uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Telemática

2018-2019

Trabajo Fin de Grado

"Machine Learning para Comunicaciones"

Sebastián Rosales Magallares

Tutor

Javier Céspedes Martín

Leganés, 2019

RESUMEN

El presente trabajo corresponde a una prueba de concepto para determinar la influencia del Machine Learning en las comunicaciones. El estudio propone una forma alternativa a los sistemas de modulación empleados en la actualidad, a partir del uso de una red neuronal para determinar la maximización de la constelación utilizada (en este proyecto 4-QAM y 16-QAM), aplicando para ello un nuevo criterio en lugar de la habitual minimización de la SER.

Este nuevo criterio implica la utilización de distancias euclídeas (método tradicional) y de distancias ponderadas, basadas en la distancia Hamming. Por tanto, se pretende determinar si, mediante el empleo de la red neuronal, es factible la aplicación de ambas, y con qué pesos cada una (determinados por el parámetro Beta en la función de coste de la red neuronal), para lograr unos rendimientos en términos de SER y BER similares a los ya obtenidos en las comunicaciones, aportando así una solución alternativa.

La primera implementación no fue satisfactoria puesto que el algoritmo de parada empleado en la red neuronal estaba únicamente basado en un número máximo de iteraciones permitidas. De esta forma, en determinadas ocasiones éstas no eran suficientes para la convergencia de la red neuronal, produciendo a la salida resultados anómalos, que se apreciaban tanto en la expansión de los puntos, en las regiones de decisión y en los rendimientos antes comentados.

En cambio, la implementación final fue exitosa en todas sus ocasiones, mostrando que a partir de cierto rango en la aplicación de pesos la eficiencia obtenida es idéntica al modelo tradicional. Fue logrado a partir de un criterio de parada basado en el valor de la función de coste de la red neuronal (loss), y a un aumento dinámico del número de iteraciones máximo en función del valor de loss, y por tanto de las necesidades de cada constelación.

No obstante, los desarrollos futuros, como el incremento del orden de la constelación o una propuesta de nuevos esquemas de codificación, en la que se aumenten las dimensiones del eje de coodenadas entrando en un hipercubo; serán pruebas más determinantes para la implantación del modelo.

Palabras clave: Red neuronal; Sistemas de Telecomunicaciones; Modulación; Prestaciones; Algoritmos iterativos; Parametrización.

DEDICATORIA

Con este trabajo acaba mi andadura en la Universidad, y es por ello que quiero acordarme de toda la gente que me ha ayudado de cualquier forma a lo largo de todo este camino.

En primer lugar a mis padres y a mi hermano, pues sin ellos el día a día hubiese sido mucho más complicado. Gracias por vuestro sacrificio diario para que nunca me faltara de nada y por hacerme sentir querido y apoyado. También a mi perro Nuuk, pues junto a él he escrito todas estas líneas y ha estudiado conmigo cada día durante la mayor parte de la carrera.

En segundo lugar a mi abuela y a mis tíos, por suponer un oasis de respiro y tranquilidad alejados de las exigencias que supone la Universidad, y por poder contar con ellos para cualquier cosa.

También quiero acordarme de mis amigos y compañeros de clase, pues son muchas las horas pasadas juntos y los momentos vividos, tanto buenos como malos; y es por ello que sois en gran medida parte de mi recuerdo de la Universidad.

Finalmente a mi tutor Javier Céspedes, por introducirme al mundo de las redes neuronales y las Telecomunicaciones. Tu pasión por el tema me hizo aceptar el desafío, además de que siempre te he tenido de ayuda y apoyo en los momentos en los que parecía que no lo íbamos a conseguir.

Gracias de corazón a todos.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

1.1. Motivación del trabajo

La principal motivación de este Trabajo Fin de Grado es la búsqueda de modulaciones más eficientes para las comunicaciones. La idea, basada en el empleo de una red neuronal, tiene como objetivo intentar obtener mejores rendimientos que los tradicionales a partir de su empleo.

Un punto clave del proyecto, y por el cual se comenzó a trabajar en él, fue la capacidad que tienen las redes neuronales de evolucionar a partir de una determinada función. Para ello, el sistema realiza una fase de entrenamiento a partir de unos datos de entrada y de forma autónoma evoluciona su resultado, devolviendo al final del proceso la constelación definitiva.

El otro aspecto importante del trabajo trata de demostrar la posibilidad de obtener resultados comparables, o incluso mejores, que los ofrecidos por los sistemas de comunicaciones ya implantados y estandarizados.

Si bien dichos sistemas basan su criterio únicamente en una minimización del Symbol Error Rate (SER), lo que implica que se tienen en cuenta exclusivamente distancias euclídeas, este proyecto trata de lograr un equilibrio entre este modelo implantado y otro basado en las distancias ponderadas entre símbolos. De esta manera, el trabajo supone una investigación para concluir si es posible un cambio de sistema o no.

Para llevarlo a cabo se planteó la posibilidad de determinar algún tipo de variable que fuera la encargada de determinar la importancia y peso otorgado a cada modelo durante la ejecución del sistema. Este parámetro, junto con otros que determinan el entorno del proyecto, será introducido por el usuario.

Para ello, la idea radica en establecer un nuevo criterio que vaya balanceando la elección de sistema desde uno basado únicamente en las distancias ponderadas entre símbolos, hacia el criterio ya implantado en la actualidad, implementado también.

En los resultados se muestra un estudio para la selección de dicho parámetro de entrada en las futuras aplicaciones de este modelo teórico, en función de los rendimientos advertidos en este trabajo.

De esta forma, se plantea la recopilación de datos, para su posterior análisis, provenientes de la ejecución de la red neuronal. Tras esto, se podrá lograr alcanzar una conclusión frente a la propuesta de introducir una nueva forma de modulación.

En caso de que los resultados fueran exitosos frente a la motivación que impulsó el proyecto, se determinaría como una prueba de concepto aprobada con el objetivo de abordar en futuros trabajos constelaciones de mayores dimensiones. Este hecho no implica

literalmente que mediante la aplicación de redes neuronales se obtuvieran siempre comunicaciones más eficientes, pero sí serviría de base para que en dichas constelaciones, empleadas en las comunicaciones, se pudieran encontrar soluciones alternativas para mejorar las prestaciones.

No obstante, si el proyecto demostrara que no es posible la obtención de mejores rendimientos frente a los ya ofrecidos, el trabajo serviría para demostrar que la mejora en las distancias ponderadas a través de la distancia Hamming no afectaría positivamente al desarrollo del Modulador. En conclusión, permitiría afirmar que cualquier empeoramiento en las condiciones tradicionales de minimización de la SER, para en cambio potenciar distancias ponderadas, sería perjudicial para el rendimiento global de la transmisión.

Por tanto, este trabajo supone una investigación relevante al tratar de mejorar las condiciones en la transmisión de las comunicaciones. Este hecho supondría permitir unas comunicaciones de mayor capacidad y rapidez, un hito cada vez más demandado en las sociedades actuales.

Para contextualizar de manera adecuada el proyecto, se detallan algunos conceptos básicos sobre modulación y redes neuronales. Toda esta información relativa al entorno global en el que se desarrolla el proyecto está explicada en el capítulo 2. Además, en este capítulo, se lleva a cabo un comentario en relación al marco regulador aplicable, así como la posible defensa de la propiedad intelectual de la idea, que se podrá encontrar en la sección 2.7. También, en la sección 2.8, se especificará el posible impacto que supondría la implementación del modelo propuesto en este proyecto.

Por último, se tratará el proyecto a fondo, detallando las distintas propuestas (capítulos 3 y 6) que se fueron realizando, sus diversos resultados y análisis posteriores (capítulos 5 y 7, así como posibles líneas de trabajo y desarrollo futuro (capítulo 8). Además, en el capítulo 7, se detallará el presupuesto empleado en el proyecto junto con los tiempos necesitados.

1.2. Resultados esperables del trabajo

Previo a la implementación del proyecto, se concluyó que este modelo será de utilidad en constelaciones que sean de tipo cuadrado, excluyendo por ejemplo constelaciones 8-QAM o 32-QAM. De esta forma serán más manejables los cálculos de distancias entre símbolos, y aplicables los métodos expuestos en este trabajo a constelaciones de rango superior.

A primera vista, se deduce que en constelaciones de pocos símbolos y mucha distancia entre ellos, las diferencias entre esta propuesta y el modelo tradicional serán prácticamente inapreciables. Sin embargo, a medida que haya más símbolos en la constelación a enviar, y por ende, menor distancia entre ellos; el empleo de la red neuronal para maximizar la distancia Hamming y la euclídea, puede suponer una mejora en el rendimiento ofrecido por el sistema.

De esta forma, se espera lograr un rango de valores para los cuales la calidad de las comunicaciones sea prácticamente igual a la del modulador clásico. Por tanto, se obtendrán un conjunto de sistemas capaces de conseguir las mismas respuestas que los tradicionales a la vez que se localizarían determinados puntos ordenados por codificación Gray.

Se decidió por capacidad de procesamiento y cómputo, llevar este proyecto hasta constelaciones de 16 símbolos, donde se supone que se verán ya muestras de la funcionalidad del Machine Learning. Sin embargo, este modelo particular estaría abierto y preparado para que en un futuro se pudiese emplear en constelaciones mayores, donde ahí sí que se esperarían obtener unos resultados más contundentes con la motivación del proyecto.

En conclusión, si el proyecto resultaba en concordancia con las teorías previas, los resultados esperados serían unos rendimientos en términos de SER y BER casi idénticos, si no superiores, que los del modelo tradicional. Por tanto, en las regiones de decisión de los símbolos se apreciarían similitudes con el estilo de rejilla o celda habitual.

Por el contrario, si tras el despliegue del trabajo, la teoría de la aplicación del Machine Learning no fuera apropiada, las regiones de decisión cambiarían de forma impredecible. Este hecho supondría, de facto, unos rendimientos totales por debajo de lo estandarizado, refutando la teoría de desarrollar un nuevo esquema de modulación.

2. ESTADO DEL ARTE

En este capítulo se tratarán los diferentes aspectos teóricos en los que se basa el proyecto. Para ello se comentarán dichos conceptos, dentro de un marco globalizado de estudio y teorías aplicables en las distintas formas de comunicación y empleo de redes neuronales.

2.1. Conceptos Básicos Previos

La transmisión de las comunicaciones implica el transporte de información desde un emisor hasta un receptor a través de un canal definido. Dependiendo de las condiciones del medio, o de las necesidades de dicha comunicación, la información será transportada de diversas formas en la onda portadora. Este conjunto de técnicas aplicadas a esa difusión de información se conoce como *Modulación* [1].

Cada pedazo de información transmitida representada se conoce como *símbolo*. Dado que la información es un elemento binario, cada símbolo es la representación en el plano de una de todas las combinaciones posibles de los bits necesarios para mandar la información. Por tanto, se establece un alfabeto que define las distintas equivalencias entre bits y su símbolo asociado. Un ejemplo podría ser la electrónica digital, en la que

$$\begin{aligned} bit = 0 \rightarrow S_0 = 0[V] \\ bit = 1 \rightarrow S_1 = 5[V] \end{aligned} \tag{2.1}$$

Una constelación muestra de forma gráfica los símbolos en el plano de coordenadas. Generalmente, en Modulación se conocen como ejes de fase y cuadratura. De esta forma, dependiendo de la modulación seleccionada, las señales transmitidas se ubicarán en el plano de una forma u otra, dando lugar a diversas formas geométricas (cuadrado, cruz...)[2].

2.2. Otros Esquemas de Modulación

En este apartado se resumen otros tipos de modulación no empleados en el trabajo, explicando brevemente sus características y las virtudes de cada uno de forma general, además de una exposición del por qué de su rechazo para la implementación del proyecto. Posteriormente se tratarán los esquemas de modulaciones de amplitud, sobre los que se basa este Trabajo Fin de Grado.

2.2.1. Modulación Angular

La modulación angular es aquella en la que la información transmitida, A[n], se transporta en aspectos angulares de la onda, ya sea en la fase o en la frecuencia. Es por ello que tradicionalmente, dentro de este esquema de modulación, se encuentran dos grandes grupos de moduladores, los de fase y los de desplazamiento de frecuencia. Es un modelo adecuado para transmitir cuando existen grandes distorsiones de amplitud. Sin embargo, como contrapunto, en condiciones más normales necesitan un mayor ancho de banda para hacerlo respecto a los modelos basados en amplitud.

Modulación Angular de Fase

Dentro de este apartado se pueden difereciar dos subtipos, la modulación de fase lineal y la de fase diferencial. En la fase lineal se encuentra el esquema PSK (Phase-Shift Keying), donde la información viaja en la fase de los símbolos. Una ventaja de este esquema es que trabaja en un rango que no satura, al hacerlo en la zona lineal. No obstante, consume un ancho de banda elevado al haber saltos de fase en t=nT.

Mejorando este esquema lineal se observa el QPSK (Quadrature Phase-Shift Keying), en donde los saltos de fase dependen de la simultaneidad (180°), o no (±90°), de cambio en las componentes de fase y cuadratura de la señal. Como colofón a la modulación de fase lineal aparece el tipo OQPSK(Offset Quadrature Phase-Shift Keying), la cual se trata de un esquema QPSK con desplazamiento temporal. De esta forma se evita que coincidan las transiciones de las componentes de fase y cuadratura, mediante el retardo de T/2 de esta última. Así se consiguen saltos únicamente de ±90° y más frecuentes (cada T/2).

Para la recepción de la transmisión de estos moduladores de fase lineales se emplean receptores PSK, u OQPSK (respectivamente), de tipo coherente (excesivamente caros), o de tipo no coherente. Con estos últimos se recibe la constelación rotada θ radianes, lo que lo hace un receptor más barato al verse posiblemente su rendimiento afectado por este hecho.

Por eso, la solución a implementar suele ser un Modulador de Fase Diferencial (DPSK), el cual obtiene unas mayores prestaciones que un receptor PSK convencional y no tiene la necesidad de emplear uno coherente. Para lograrlo, emplea una codificación diferencial y una realimentación en su esquema de modulación.

Modulación Angular por Desplazamiento de Frecuencia

Conocida como modulación FSK (Frequency-Shift Keying), se caracteriza porque la información viaja en los pulsos de frecuencia discreta de las portadoras. De esta forma se generan M pulsos para M símbolos a transmitir.

En un avance de este esquema se obtiene la modulación CPFSK (Continuous Phase

Frequency-Shift Keying), que, como su propio nombre indica, se trata de una FSK de fase continua, lo que implica que los pulsos tengan un número entero de periodos en T segundos.

Finalmente, el último grupo de modulación dentro del Angular por Desplazamiento de Frecuencia es el MSK (Minimum-Shift Keying), el cual, a diferencia de los FSK, su información se traslada en los cambios de frecuencia en la frecuencia de las portadoras. Con esta modulación se consigue que exista una mínima separación de frecuencia al emplear portadoras ortogonales entre sí.

Como conclusión, debido a que el desarrollo del proyecto se lleva a cabo en un entorno que no es hostil en las distorsiones de amplitud, esta forma de modulación se descarta, puesto que supone introducir un déficit añadido a los rendimientos que se obtengan, debido al mayor ancho de banda que necesitan estos tipos de modulación [3].

2.2.2. Modulaciones Multipulso

Modulaciones de Espectro Ensanchado (SS)

Mediante este esquema, se obtiene un ancho de banda deliberadamente mayor mediante el empleo de un factor de aumento N. De esta manera se tratan de evitar los desvanecimientos, consiguiendo así una mayor protección frente a interferencias [4].

Está pensado para comunicaciones que necesiten más seguridad (tuvo un origen militar al querer paliar el jamming) y permite baja sensibilidad frente a las distorsiones producidas por el canal. Sin embargo, un mito ya desmentido es que pudiese incrementar la capacidad del sistema, lo cual es rotundamente falso.

Dentro de esta categoría se encuentran dos subtipos, DS-SS (Direct Sequence SS) y CDMA (Code Division Multiple Access).

La primera de ellas, aunque ya casi está descartada en la actualidad, se empleaba para reducir las interferencias gracias a la combinación lineal de diversas réplicas de un mismo pulso.

No obstante la forma multipulso más empleada hoy en día es CDMA. Este mecanismo permite que múltiples usuarios tengan acceso a la misma banda de frecuencias al mismo tiempo. Gracias a que cada uno de ellos posee un código de usuario diferente, se posibilita la utilización de una secuencia de ensanchado distinta para cada uno. Además, para una optimización del sistema, se emplean pulsos ortogonales. Por tanto, se obtiene la mínima separación posible entre las señales de cada usuario, maximizando de esta manera el número total de usuarios que pueden utilizar el sistema simultáneamente.

A pesar de ello, esta forma de modulación queda descartada en el proyecto debido a que la capacidad del canal ante el ruido es inferior que empleando modulaciones de amplitud. Además, puede sufrir los efectos de las interferencias de múltiple acceso, pudiéndose

producir de esta forma grandes errores en las estimaciones de símbolos. Por último, CDMA suele implicar el empleo de transmisores más veloces, lo que implica mayor coste [5].

2.3. Nuestro Esquema de Modulación: Modulaciones de Amplitud

2.3.1. Modulación por Amplitud de Pulsos (PAM)

Esta técnica de modulación se basa en la variación de la amplitud de las señales transmitidas [6], es decir, las amplitudes del tren de pulsos dependen de los valores de la señal original. Por tanto, permite la transmisión de una única señal, cuyas regiones de decisión serán los puntos medios entre símbolos a lo largo del eje real [7].

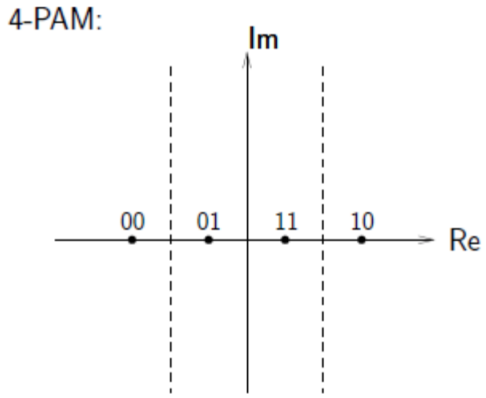

Fig. 2.1. Ejemplo Constelación 4-PAM.

2.3.2. Modulación de Amplitud de Cuadratura (QAM)

Esta técnica (en su versión analógica), en contraposición al modelo PAM, supone el transporte de dos señales independientes de una misma portadora, moduladas en amplitud y fase. Por tanto, este tipo de modulación se basa en la suma de ambas señales moduladas.

Sin embargo, el proyecto se lleva a cabo en su versión digital, conocida como QAM-*Cuantizada*. En esta ocasión, a partir de datos binarios, se forman grupos de tantos bits como sean necesarios (m) para obtener los N estados modulados diferentes, gracias a los cambios en la fase y amplitud de la portadora.

De esta forma, para las constelaciones N-QAM, se establecerán

$$2^m = N \tag{2.2}$$

estados diferentes, y para representarlos, se recurre a las constelaciones, explicadas en la sección 2.1. Debido al uso de bits, sus órdenes de constelación (número de símbolos) son potencias de 2 y pueden tener un aspecto cuadrado(4-QAM,16-QAM,64-QAM... conformados sus puntos en forma de rejilla), u otros aspectos diferentes como las constelaciones 8-QAM o 32-QAM.

Este modelo es seleccionado para la implementación del trabajo frente al modelo PAM debido a su mayor eficiencia. Gracias a esto se puede incrementar en gran medida el orden de la constelación a transmitir (1024-QAM...). Por ello, es el modelo empleado en las comunicaciones en la actualidad (TDT, TV por cable, módems...) [8].

2.4. Red Neuronal Clásica: El Perceptrón Simple

Una red neuronal, artificial, es un concepto (Hebb Learning) que se desarrolló durante el S.XX [9] con motivo de intentar reproducir el comportamiento de las neuronas humanas. De esta forma, estos sistemas deben ser capaces de sufrir cambios y evolucionar a través de un aprendizaje automático. La red estará formada por diversas unidades, neuronas, a las cuales les llega una información de entrada (inputs), que será procesada (se operarán los datos de entrada) y finalmente devolverán los valores salientes (outputs) al cumplirse un *criterio de parada*.

Cada neurona está conectada a la siguiente mediante enlaces (Fig.2.2), los cuales tienen un peso específico, que multiplicará el valor de salida de la neurona anterior. Además, a la salida de una neurona puede existir una *Función de Activación*, que modificará el valor obtenido, actuando como umbral o limitador (no propagará la información a la siguiente neurona hasta que su valor se haya sobrepasado) [10] [11].

Por ello, la regla general para la implementación de redes neuronales es:

$$outputs = activation(inputs \times kernel + bias) \tag{2.3}$$

siendo el *kernel* los n filtros utilizados por la red neuronal y *bias* un parámetro adicional de la red neuronal que proporciona mayor grado de libertad al sistema.

La evolución y aprendizaje de la red neuronal se consigue gracias a su entrenamiento. Por tanto, consiste en que la red aprenda el comportamiento que se esconde en dichos datos de entrada, mediante la minimización de una determinada *función de coste*. A partir de los inputs (etiquetados para guiar la función de coste y el entrenamiento) y de los outputs, se formará dicha función de coste:

$$F.Coste = f(outputs, etiquetas) \tag{2.4}$$

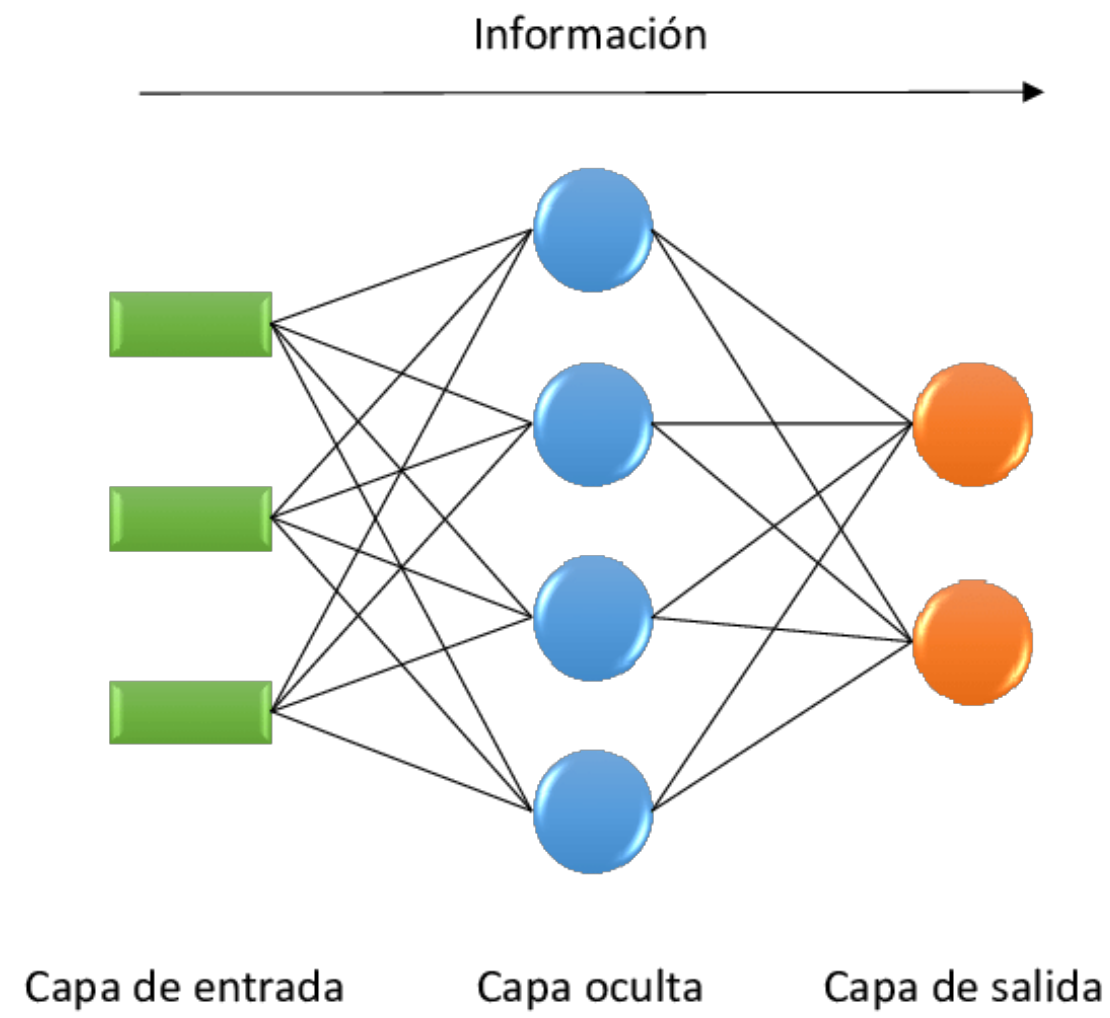

Fig. 2.2. Concepto de Red Neuronal Artificial [12].

Aplicando este concepto, Frank Rosenblatt (1958)[13], diseñó lo que denominó como Perceptrón Simple (Fig. 2.3). Se trata del modelo más simple de red neuronal, puesto que el sistema se compone solamente de una capa. Mediante una simple función se define la función de activación empleada:

$$f(x) = \begin{cases} 1 & \text{si } w \times x - u > 0 \\ 0 & \text{en otro caso} \end{cases} \tag{2.5}$$

siendo 'u' el umbral de dicha función de activación. Una vez establecida la regla de actuación de la neurona, es necesario establecer cómo debe evolucionar, es decir, aprender de forma automática. Para ello se aplica el concepto de la Teoría Hebbiana [11], el cual hará ir modificando los pesos establecidos conforme a un *factor de aprendizaje* (α, constante entre 0 y 1) y a la salida deseada (δ), en función de la iteración del sistema (j) y

del vector de entrada (x).

$$w(j)' = w(j) + \alpha(\delta - y)x(j) \tag{2.6}$$

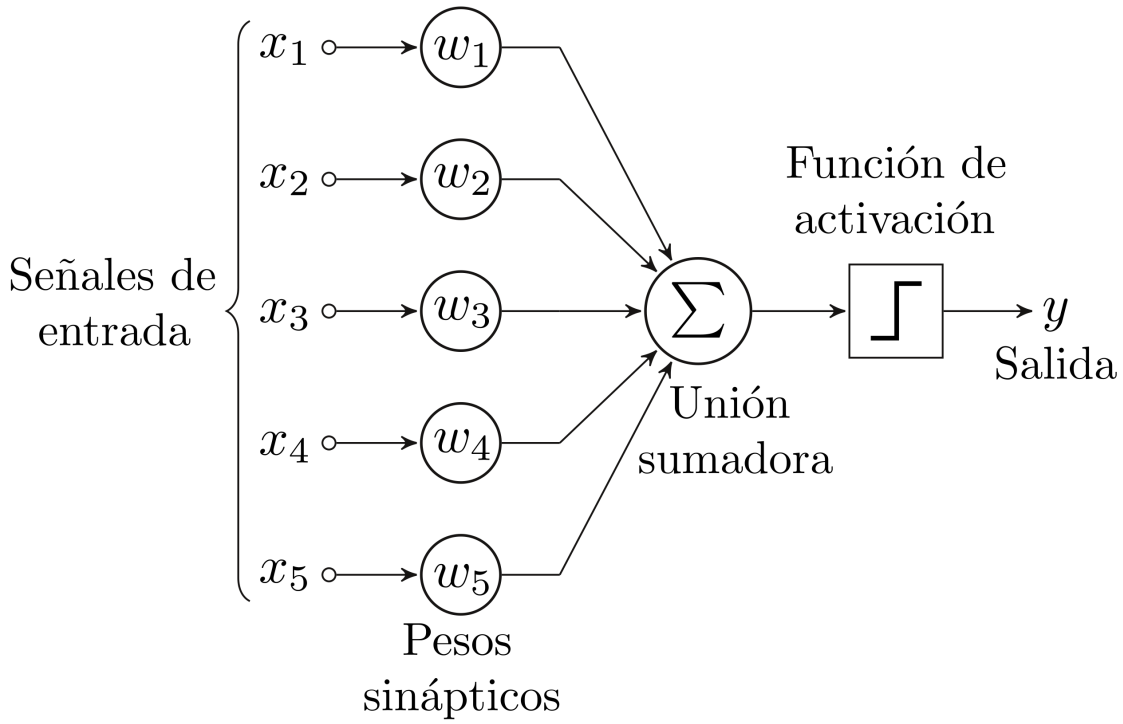

Fig. 2.3. Perceptrón Simple [14].

No obstante, pronto se catalogó este sistema como inútil ya que "solo sirve clasificar problemas linealmente separables, cosa que ya se podía hacer mediante métodos estadísticos, y de una forma mucho más eficiente"[9].

2.5. Nuestro modelo de Red Neuronal: El Perceptrón Multicapa

A pesar del nulo empleo del Perceptrón Simple, la idea original continuó siendo desarrollada hasta alcanzar el modelo que se empleará en este trabajo, el Perceptrón Multicapa. Definido en 1986, dentro de los estudios conocidos como *Paralell Distributed Processing* (PDP)[15], se entiende que es una continuación del Perceptrón Simple. Algunas de sus características son:

- Introducción de una o más capas intermedias.
- Se permite que la entrada sea continua.
- Las capas intermedias suelen tener una activación de tipo sigmoide, mientras que la final suele tenerla lineal.

En este caso, al haber un mayor número de capas, el entrenamiento de la red será más complicado, puesto que cada capa tiene una matriz W_k y un vector b_k (2.3). De esta forma, los símbolos transmitidos serán función de:

$$outputs = f_k(W_k \times f_{k-1}(W_{k-1} \times ... \times f_2(W_2 \times f_1(W_1 \times bits + b_1) + b_2)... + b_{k-1}) + b_k) \quad (2.7)$$

No obstante, su principal diferencia frente al modelo simple, y que supone su mayor ventaja es la modificación del sistema de aprendizaje de la red. El nuevo Perceptrón incorpora un *algoritmo de retropropagación* (Backpropagation-BP)[16], basado en:

$$\Delta w_{i_j} = \epsilon \times \delta_i(t) \times a_j(t) \quad (2.8)$$

en donde

- "$a_j(t)$ es la salida del elemento de proceso j"[9].
- "$\delta_i(t)$ es la derivada de la señal de error en la capa de salida con relación a la función de activación en el elemento de proceso i (que es, como en el perceptrón simple, el producto escalar de vector de entrada por vector de pesos)"[9].

Este tipo de red neurnoal tiene como característica que el tamaño introducido a la entrada de las capas intermedias, y por ende, el tamaño de las salidas de éstas intermedias, es definible por el programador. Por eso, el tamaño inicial introducido a la entrada del sistema, y el final producido a su salida, son independientes de los de las capas intermedias.

Además, esta nueva implementación de red neuronal tiene la capacidad de gestionar diversos problemas que surjen en tiempo de ejecución, tales como el *sobreajuste* (número innecesario de neuronas en capas intermedias) o el *sobreentrenamiento*.

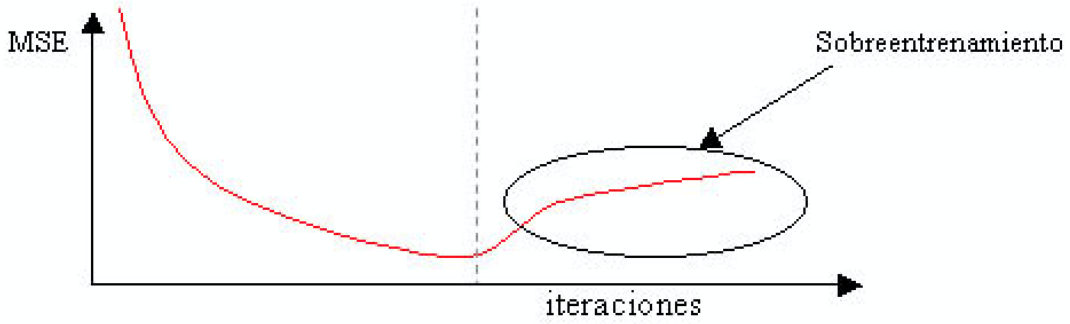

Fig. 2.4. Problema de Sobreentrenamiento de la Red Neuronal [9].

2.6. Receptor de Máxima Verosimilitud (ML)

Tras ser la información a transmitir modulada, los símbolos se envían a través del canal. La recepción será de la forma:

$$r(t) = s(t) * h_c(t) + w(t) \quad (2.9)$$

en donde los diferentes elementos son:

- Señal donde se transmiten los símbolos: $s(t)$
- Canal Discreto Equivalente: $h_c(t) = \delta(t)$
- Ruido introducido al canal: $w(t)$. Será del tipo aditivo blanco gaussiano (AWGN), con una Densidad Espectral de Potencia σ_n^2

Tras la modulación y la transmisión de los símbolos a través del canal, es necesario establecer un decisor en el receptor que dirima los símbolos que se transmitieron. Para ello se emplea el receptor de *máxima verosimilitud*. Al ser todos los símbolos equiprobables en este proyecto (este decisor coincide en este caso con el de Máxima a posteriori-MAP [17]), aplica un criterio de mínima distancia, puesto que su decisión depende únicamente de seleccionar la mínima distancia euclídea entre la observación recibida y los diferentes símbolos de la constelación:

$$\hat{B} = arg \min_i dist(s_i, r) \tag{2.10}$$

siendo $\hat{B}$ los diferentes símbolos estimados e

$$i \in |M|$$

Para calcular dicha distancia euclídea entre dos puntos en el plano se aplica:

$$Punto_1 = (x_1, y_1) \tag{2.11}$$

$$Punto_2 = (x_2, y_2) \tag{2.12}$$

$$Distancia(Punto_1, Punto_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2.13}$$

No obstante, si no fueran equiprobables, la minimización de la distancia entre los símbolos de la constelación no sería el criterio a tener en cuenta para su decisión.

2.7. Marco Regulador

Este trabajo se considera una prueba de concepto para concluir si existe la posibilidad del empleo de redes neuronales en las comunicaciones, así como para establecer una diferenciación frente al modelo tradicional basado en la minimización de la SER.

Por tanto, al ser una prueba inicial como punto de partida de desarrollos futuros, esta posible tecnología no está dentro de ningún marco regulatorio como tal. Si bien es posible que en el futuro se plantee la posibilidad de introducir patentes para proteger la propiedad

intelectual, en la actualidad se trata de una fase tan experimental del sistema que no es recomendable su intento de aplicación.

No obstante, la base de modulación sobre la que se sustenta el trabajo (QAM), si que se puede encontrar en diversas aplicaciones y estándares. Es empleado en aplicaciones *broadcast* a usuarios, como la televisión o los módems por cable. Por ejemplo, en Estados Unidos se emplean las constelaciones 64-QAM y 256-QAM como esquemas para la televisión por cable bajo el estándar *ANSI/SCTE07 2000*. Además es usado en múltiples aplicaciones móviles, mediante tecnologías inalámbricas y celulares, como por ejemplo en la implantación del 5G a través de antenas de microondas a partir de constelaciones 16-QAM [18], o en el uso de constelaciones 1024-QAM para el LTE-Downlink (E-UTRA, Technical Report, Release 15).

2.8. Impacto Socio-Económico

Como se comentaba anteriormente, al encontrarse el trabajo en una fase beta de desarrollo, su implantación dentro del mundo comercial tardará en llegar. Tras la evaluación de las múltiples pruebas futuras, será necesario encontrar un nicho de mercado lo suficientemente subevolucionado en el que pueda interesar la aplicación de esta metodología, puesto que en los sectores más desarrollados un cambio de tecnología supondría cambiar muchos equipos con su correspondiente coste.

Si bien es cierto que el uso de la modulación QAM está muy extendido en las comunicaciones, ya sea por cable o a través de redes celulares, su aplicación está ligada al modelo tradicional de decisión. Por tanto, las pruebas futuras deberían determinar un avance significativo de la tecnología expuesta en este proyecto frente a lo implantado para lograr un cambio de modelo.

Este hecho sí que podría suponer un gran impacto debido a las múltiples aplicaciones, con infinidad de usuarios, que emplean dichas tecnologías de comunicación.

3. PRIMERA PROPUESTA

3.1. Planteamiento inicial

La base inicial del proyecto se basa en la aplicación del Machine Learning como elemento de iniciación aleatorio, pero con la capacidad de evolucionar hacia los intereses introducidos en el capítulo 1.

La idea es que todos los puntos de la constelación partan aproximadamente del origen de coordenadas, y que sea la red neuronal la que, iteración tras iteración, guíe a cada punto hacia la zona más conveniente con el fin de maximizar las prestaciones.

En primer lugar, para conseguirlo es necesario transformar los bits que llegan a la entrada de la red neuronal a los puntos finales de la constelación que se muestran. Para ello se definen unas *variables de entrenamiento*, que serán las que vayan modificándose posteriormente durante la fase de entrenamiento (introducida en el cap.1), mediante los distintos pesos que vayan adquiriendo y la evolución del aprendizaje autónomo.

Una vez definidas estas variables, se establecen las operaciones que se deben llevar a cabo para los cálculos de las distancias euclídea y ponderada. En el caso de distancias euclídeas, tal y como se señaló en la ecuación 2.13, se obtendrán directamente las distancias entre todos los puntos simplemente con su aplicación, siendo su mínima el valor más pequeño dentro de la matriz de distancias resultante. Puesto que las distancias

$$dist(Punto_1, Punto_2) = dist(Punto_2, Punto_1) \tag{3.1}$$

son iguales, esta matriz obtenida será simétrica por la diagonal.

Por el contrario, el cálculo de las distancias ponderadas implica un proceso diferente, puesto que influye el orden de la constelación y la distancia hamming (número de bits de diferencia entre ambos tras comprobar la relación bits-símbolo) que haya entre ambos puntos, así como su propia distancia euclídea. Esto supone que teniendo dos distancias euclídeas iguales tendrá menor distancia ponderada aquella que tenga mayor distancia hamming.

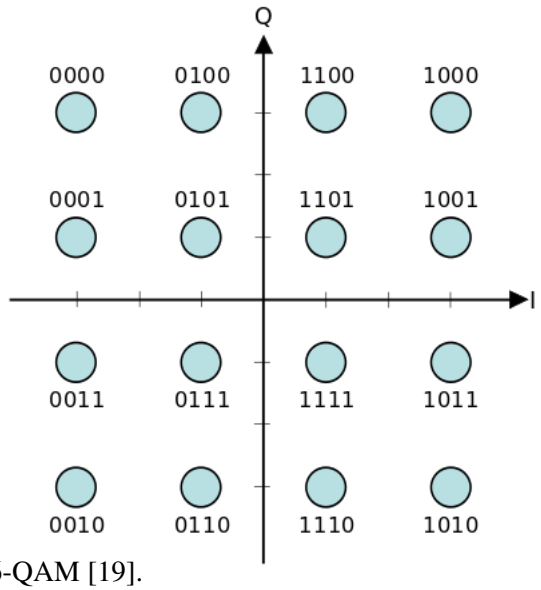

Fig. 3.1. Constelación 16-QAM [19].

A modo de ejemplo, los bits de información 0000 son modulados como (-3,3) y 0001 como (-3,1). Su distancia hamming será:

$$dist.Hamming = 1 \tag{3.2}$$

puesto que sólo hay un bit de diferencia entre ellos. Por otro lado, su distancia ponderada empleará la fórmula de la distancia euclídea (2.13), pero dividiendo ésta por la distancia hamming obtenida, dando como resultado:

$$dist.Ponderada = \frac{dist.Euclidea}{dist.Hamming} = \frac{2}{1} \tag{3.3}$$

Tras la obtención de todas las distancias pertinentes, es preciso que el sistema sepa relacionar ambas de forma adecuada. Esta tarea es llevada a cabo por la función de coste (2.4) definida. Debido a que uno de los objetivos del proyecto es relacionar ambas distancias mínimas, la función de coste deberá hacerlo, otorgándole un peso dinámico a cada una.

El valor de dicho peso lo representará una variable denominada **Beta**. Por tanto, esta variable tomará valores en el rango [0,1], y en función de su valor, le dará mayor importancia a la distancia mínima ponderada (propuesta del proyecto) o a la distancia mínima euclídea (modelo estandarizado). Por ello, la función de coste quedará de la siguiente forma:

$$F.Coste = Beta \times minEuclidea + (1 - Beta) \times minPonderada \tag{3.4}$$

Como se puede apreciar, la función de coste es una suma de distancias mínimas dependiente del peso otorgado a la variable. El siguiente paso consiste en modificar las variables de entrenamiento para la evolución de la red neuronal, mediante la minimización de dicha función de coste. (2.4)Para conseguirlo se calcula un gradiente de tal manera que se

obtiene cómo se tienen que modificar las variables de entrenamiento de cara al siguiente paso en el algoritmo iterativo.

Este proceso explicado, desde la conversión de bits a símbolos hasta la modificación de las variables de entrenamiento, se lleva a cabo durante cada iteración que necesite la red neuronal antes de toparse con su criterio de parada. (2.4)

3.2. Criterio de Parada

Algoritmo basado en iteraciones

En esta primera aproximación al planteamiento no se plantea ningún algoritmo específico para la parada de la red neuronal, ya que tampoco se sabe con seguridad cómo va a responder esta.

Es por ello que se da un número de 10.000 iteraciones, pasadas las cuales el sistema dejará de ejecutarse y se obtendrán los datos resultantes. El por qué de esta cifra es sencillo de explicar, se piensa que con este número de iteraciones la constelación ya habrá sido capaz de expandirse completamente, por lo que se cortará la ejecución directamente.

A primera vista se intuye que los puntos se expandirán marchando directamente hacia su punto de maximización en línea prácticamente recta sin trazar curvas ni dar vueltas. Esta posibilidad parece intuitiva a priori, pero falta llevar a cabo las simulaciones para cerciorarse completamente. Es por eso que no se presta mayor importancia al algoritmo de parada en primera instancia, sin buscar algo más empírico, pues ese número de iteraciones dentro de la red neuronal parecen ser más que suficientes para lograr su convergencia.

Sin embargo, si el sistema ha logrado su maximización antes de concluir el número de iteraciones, seguirá ejecutándose hasta que concluyan éstas.

4. PRIMERA IMPLEMENTACIÓN

4.1. Implementación de la Red Neuronal

La implementación de la red neuronal se basa en la utilización de la librería TensorFlow. Este API de código libre permite la posibilidad de construir la red neuronal y entrenarla, a partir de diversos parámetros que se establecen, así como con las variables de entrenamiento.

Básicamente, para la creación y el uso de la red neuronal, se define la función *compute_graph_MODULATOR*. De esta función, con unas características especiales que se detallarán posteriormente, se ramifican dos funciones necesarias también para el correcto funcionamiento de la red neuronal, las cuales son *Mod* y *pdist_*.

Para la implementación de la red neuronal se emplea el lenguaje de programación *Python*, puesto que permite un fácil manejo de TensorFlow, y por tanto, un desarrollo más intuitivo de la propuesta.

La función compute_graph_MODULATOR

Tal y como se explicó la sección 2.4, a la red neuronal llegan unos inputs, que serán procesados de forma autónoma dentro del sistema y finalmente se devolverá una salida determinada. Este hecho supone que la red neuronal actúe como una caja negra, en la que el programador detalla una serie de instrucciones que formulan las operaciones que utilizará la red neuronal. Esta tarea es llevada a cabo por la función *compute_graph_MODULATOR*. En ella se definen un conjunto de instrucciones, y la librería TensorFlow es la encargada de aportarles valor durante la ejecución del sistema.

Por este motivo, y aunque para la implementación del conjunto del sistema son necesarias otras funciones auxiliares, esta función ocupa el escalón más alto en la jerarquía. En ella se conforma el contorno de la red neuronal a partir de aspectos básicos introducidos por el programador, empleando dichas funciones auxiliares para momentos concretos de la ejecución.

Estos datos de entrada necesarios para la red neuronal se obtienen como parámetros de la función, los cuales son en su mayoría información básica del funcionamiento del sistema:

- El orden de la constelación (batch_size).
- Bits necesarios para codificar un símbolo (k).
- El número de ejes (eje real e imaginario) (n).

- El número de capas intermedias de la red neuronal (en este caso 1, a la que se añadirá la capa final) (cod_additional_density).
- Las no linealidades de cada capa (cod_activation_add).
- La varianza con la que poder inicializar aleatoriamente (alpha).
- Parámetro de actualización del kernel (lr).

Una vez establecidos los parámetros de entrada, ya se puede definir el mecanismo apropiado de TensorFlow sobre el que se implementarán las instrucciones, el tensor *Graph*. De esta forma se define y se detallan las instrucciones a llevar a cabo posteriormente, tal y como se muestra a continuación.

```
graph = tf.Graph() # simula una caja negra
 with graph.as_default(): # A continuacion las instrucciones
```

Para cumplir con la propuesta del proyecto establecida, el primer aspecto a tratar debe ser el paso de bits de entrada a símbolos. Para ello es necesario en primer lugar reservar memoria para el alfabeto (combinaciones bits-símbolo) que se recibirá, siendo ésta la primera instrucción del graph.

```
  M=tf.placeholder(tf.float64,shape=(batch_size,k),name='bits')
```

Con el espacio reservado, ya es posible definir las variables de entrenamiento. Pero antes, en TensorFlow es necesario otorgarlas un nombre con el que poder discernir qué instrucciones deben seguir cada conjunto de variables de entrenamiento (para el supuesto caso de que hubiera más).

```
with tf.variable_scope("scope_cod2"):
```

Tras esto, se emplea la primera función auxiliar, **mod**, encargada de llevar a cabo las funciones propias de modulación, a partir de algunos de los parámetros de entrada de la red neuronal.

```
A=mod(M,n,cod_additional_density,cod_activation_add,alpha)
```

Dentro de esta función auxiliar se determinan diversas instrucciones, como el número máximo a codificar, la elección de la no-linealidad que se desee o la propia definición de las variables de entrenamiento.

```
to_bpsk=2*M-1
available_activations=[tf.nn.softplus,tf.nn.softsign,tf.sigmoid,tf.
   tanh]
def_init = tf.random_normal_initializer(stddev=alpha) #gaussiana
hidden=[] #lista para agregar las capas
```

Como detalle, la no-linealidad que se escogerá siempre, tanto en capas intermedias como en la final, será la tangente hiperbólica, pues permite acotar los resultados en un rango $[-1, 1]$.

A continuación ya se pueden establecer las diversas capas por las que aprenderán las variables de entrenamiento durante la ejecución, y que finalmente devolverán los símbolos de la constelación modulada. En el proyecto, al definirse únicamente una capa intermedia, ésta deberá tener como tamaño de entrada el del máximo número a codificar y como salida *units*, definido en 20.

$$output_1 = activation(W_0 \times to\_bpsk + b_0) \tag{4.1}$$

Aplicando la no-linealidad, los valores de las variables permanecen en el rango adecuado también en estas partes intermedias de la red neuronal.

```
hidden.append(tf.layers.dense(to_bpsk,units,activation=
    available_activations[cod_activation_add[it]],kernel_initializer
    =def_init))
```

En el caso de que hubiera más capas intermedias, éstas serían de la forma:

$$output_2 = activation(W_{it} \times hidden[it - 1] + b_{it}) \tag{4.2}$$

siendo $hidden[it - 1]$ la salida producida por la capa anterior, como se muestra a continuación:

```
hidden.append(tf.layers.dense(hidden[it-1],units,activation=
    available_activations[cod_activation_add[it]],kernel_initializer
    =def_init))
```

Finalmente se añade la última capa de la red, de forma similar a la capa intermedia, la cual tendrá un tamaño a su entrada igual al de la salida de la capa intermedia (units) y un tamaño de salida de 2 (número de dimensiones de la constelación requerida) Como siempre, debe introducirse la tangente hiperbólica para mantener el rango.

```
A=tf.layers.dense(hidden[-1],n,activation=activation_end,
    kernel_initializer=def_init)
```

Este tensor *A* es devuelto a *compute_graph_MODULATOR*, y permitirá durante la ejecución, obtener los diversos símbolos en el plano definido gracias al aprendizaje de las variables de entrenamiento, las cuales se recogen y almacenan en un listado tras su uso en la iteración a partir del nombre dado.

```
 var_list_cod = tf.get_collection(tf.GraphKeys.TRAINABLE_VARIABLES,"
    scope_cod2")
```

Una vez obtenidos los símbolos de la constelación en el plano, es necesario determinar las instrucciones para el cálculo tanto de la distancia euclídea como de la ponderada. Para ello se vuelve a recurrir a una función auxiliar, **pdist_**, la cual necesita 3 parámetros de entrada, el tensor recibido de la función *mod*, la variable *M* (espacio reservado en memoria para el alfabeto) y una variable auxiliar para el cálculo de la distancia mínima (*k*), sobre la que también hay que reservar espacio.

```
K=tf.placeholder(tf.float64,shape=(),name='K')
```

Por tanto, con dichos parámetros la función auxiliar devuelve, a partir de mecanismos de TensorFlow, las matrices *pairwise* de ambas distancias (en Tensores) y sus mínimos respectivos. Como aclaración, una matriz pairwise es aquella que se obtiene tras la comparación de dos entidades, en este caso puntos de la constelación, siguiendo unos criterios fijados. Por tanto, es ideal para el objetivo del trabajo porque es justamente lo que se necesita.

```
D_,minTrue,D_2,minPond=pdist_(A,M,K)
```

Tras la obtención de dichas distancias mínimas, ya se puede pasar a definir la función de coste de la red neuronal. Como se determinó en la propuesta, la función de coste debe relacionar ambas distancias mínimas, otorgando un peso determinado a cada una en función del valor de la variable **Beta** definido por el usuario. Al ser Beta el peso de una de esas distancias y estar en el rango [0, 1], la otra distancia estará ponderada por $(1-\beta)$. De esta forma, la función de coste queda:

```
loss=Beta*minTrue+(1-Beta)*minPond
```

siendo su resultado una suma del producto de distancias y pesos. Por tanto, como se puede apreciar, si Beta toma el valor de 0, únicamente se tendrán en cuenta las distancias ponderadas (sistema propuesto), mientras que si Beta valiera 1, simularía el modelo tradicional estandarizado. Para otros valores, Beta simula el balance de distancias, siendo su labor de gran importancia debido a que uno de los grandes objetivos del proyecto es determinar cómo afecta al rendimiento del sistema la mezcla de influencia de ambas distancias en la decisión.

Por último, para lograr el aprendizaje de la red, se deben modificar las variables de entrenamiento mediante la maximización de la mínima distancia, contenida en forma de factor de *loss*. Por tanto, mediante la minimización de $-loss$ se consigue dicho objetivo. Mediante la siguiente instrucción se modificarán las variables de entrenamiento minimizando el $-resultado$ de la función de coste.

```
optimizer = tf.train.AdagradOptimizer(learning_rate=lr,
    initial_accumulator_value=0.1,use_locking=False, name='Adagrad')
    .minimize(-loss)
```

Para su posterior uso, es preciso guardar los diferentes tensores obtenidos dentro de la caja negra, por lo que se guardan en un diccionario, que será devuelto al usuario, así como el propio graph.

```
graph_dictionary={"M":M,"K":K,"Beta":Beta,"A":A,"loss":loss,"
    optimizer":optimizer,"var_cod":var_list_cod,"minTrue":minTrue,"
    D_":D_,"minPond":minPond,"D_2":D_2}
```

Todos estos pasos mostrados en la implementación suponen la actuación del sistema durante una sola iteración. Por tanto, es importante determinar su evolución durante las miles de iteraciones que puede necesitar para converger así como la implementación del algoritmo de parada desarrollado.

4.2. Evolución del Modulador de la Red Neuronal

El primer paso para la evolución de la red neuronal supone definir el graph sobre el que TensorFlow opere las instrucciones previamente descritas.

```
# Definicion del graph a utilizar
[my_computed_dictionary,my_graph]=compute_graph_MODULATOR(batch_size
    ,k,n,cod_additional_density,cod_activation_add,alpha=alpha,lr=lr
    )
```

Como se puede ver, simplemente cambiando los parametros pasados a la función *compute_graph_MODULATOR*, la red neuronal trabajaría desarrollando cualquier otra constelación (dentro de las apropiadas) empleando el número de capas que se prefiriera.

El siguiente paso es inicializar la sesión y un *saver* en donde se guardarán las variables de entrenamiento, y se ejecuta el inicializador de variables globales.

```
with tf.Session(graph=my_graph) as session:
            saver = tf.train.Saver()
            tf.global_variables_initializer().run()
```

Finalmente, se ejecuta la llamada a la función, pasándole diversos valores guardados en el diccionario tras la modificación de las variables de entrenamiento al final de la iteración anterior. También recibe las variables necesarias para los cálculos de distancias y función de coste dentro del graph. Esta ejecución se lleva a cabo siempre y cuando no se llegue al número máximo de iteraciones, puesto que el criterio de parada(3.2), está basado únicamente en el número de iteraciones.

```
while (it<num_max):
```

```
fd={my_computed_dictionary['M']:messages,my_computed_dictionary['K']
   ]:K,my_computed_dictionary['Beta']:Beta}
```

```
 _,loss,minTrue,minPond,lcod,cod_tab,Ds,Dpon= session.run([
     my_computed_dictionary['optimizer'],my_computed_dictionary['
     loss'],my_computed_dictionary['minTrue'],my_computed_dictionary
     ['minPond'],my_computed_dictionary['var_cod'],
     my_computed_dictionary['A'],my_computed_dictionary['D_'],
     my_computed_dictionary['D_2']], feed_dict=fd)
```

Como se puede ver, en cada iteración se recogen diferentes variables que se emplearán cuando la red neuronal converja. Para su correcta comprensión, algunas de ellas son:

- lcod: array con los valores de kernel y bias.
- cod_tab: nuevos puntos de la constelación.
- Ds: matriz de distancias euclídeas.
- Dpon: matriz de distancias ponderadas.

4.3. Otro lenguaje de Programación: Matlab

En este lenguaje se lleva a cabo todo el análisis, analítico y gráfico de los resultados obtenidos en Python. Para ello, se estructuran tres bloques que servirán para representar los diversos aspectos a tratar (expansión de la constelación, regiones de decisión y representación de rendimientos).

En primer lugar se definió el código necesario para la representación de la expansión de puntos tras todas las iteraciones guardadas de Python.

Posteriormente se desarrolló el código con el que se calculan las tasas de error del sistema, SER y BER (Bit Error Rate). Gracias a ellas se podrán comparar los resultados obtenidos en la implementación con los tradicionales e ideales.

Finalmente se llevó a cabo el último bloque, la implementación de las regiones de decisión obtenidas y la tradicional. Se desarrolló al final debido a que surge como complemento a la representación de las constelaciones, permitiendo un mejor entendimiento de los resultados.

5. PRIMEROS RESULTADOS Y CONCLUSIONES

Utilizando el modelo de canal (2.9) y el decisor ML (2.6), se lleva a cabo un estudio para determinar el rendimiento del sistema propuesto y compararlo con la modulación normalizada.

Para poder recabar unos resultados convincentes eliminando el azar lo máximo posible, se decidió que se ejecutaría el código tres veces por cada Beta. De esta forma, se podrán comparar los resultados de las diferentes simulaciones para cerciorarse de que siempre son prácticamente los mismos.

Para llevarlos a cabo es necesario una serie de parámetros básicos, algunos de ellos definidos por el programador y otros obtenidos a partir de diversas maneras.

Los parámetros definidos son los más simples, como el orden de la constelación, el número de símbolos a transmitir (en este proyecto 10^7) y el número total de bloques de símbolos que se mandarán (1000). Por tanto, el número de símbolos transmitidos en cada bloque será el cociente entre ambos(10^4). El último parámetro que se define tal cual es el número máximo de errores (2000), que permite al sistema transmitir y recibir hasta alcanzarlo.

En la obtención del resto de parámetros, se pueden apreciar dos grandes bloques. El primero de ellos está relacionado con la obtención de los datos de interés a partir de los proporcionados por la red neuronal. Estos son:

- Número de bits por símbolo.
- Puntos de la constelación, obtenidos tras la ejecución de la red neuronal.
- De forma adicional, para comprobar la correcta importación de los datos, se obtienen a partir de éstos la energía de bit y la total de la constelación.
- Generación aleatoria de los símbolos a transmitir $[0, M-1]$, en forma de vector de tamaño el número total de símbolos definido.
- Transmisión de todos los símbolos generados, que serán modulados posteriormente por funciones específicamente desarrolladas.

El segundo bloque, de forma análoga al primero, se encarga de la transmisión y modulación, pero de la forma tradicional, en la que las funciones para la modulación son las estandarizadas.

5.1. Análisis de las gráficas

En primer lugar se comienza por simular contelaciones 4-QAM (por los motivos explicados en la sección 3.1), las cuales, como se muestra a continuación, son exitosas en las primeras pruebas en términos de expansión de la constelación, y por tanto de sus regiones de decisión y tasas de error, ya que para todos los betas se logran los mismos rendimientos que en el caso ideal y en el del modulador clásico.

A modo de ejemplo, para distintos Betas estas son sus constelaciones y regiones de decisión.

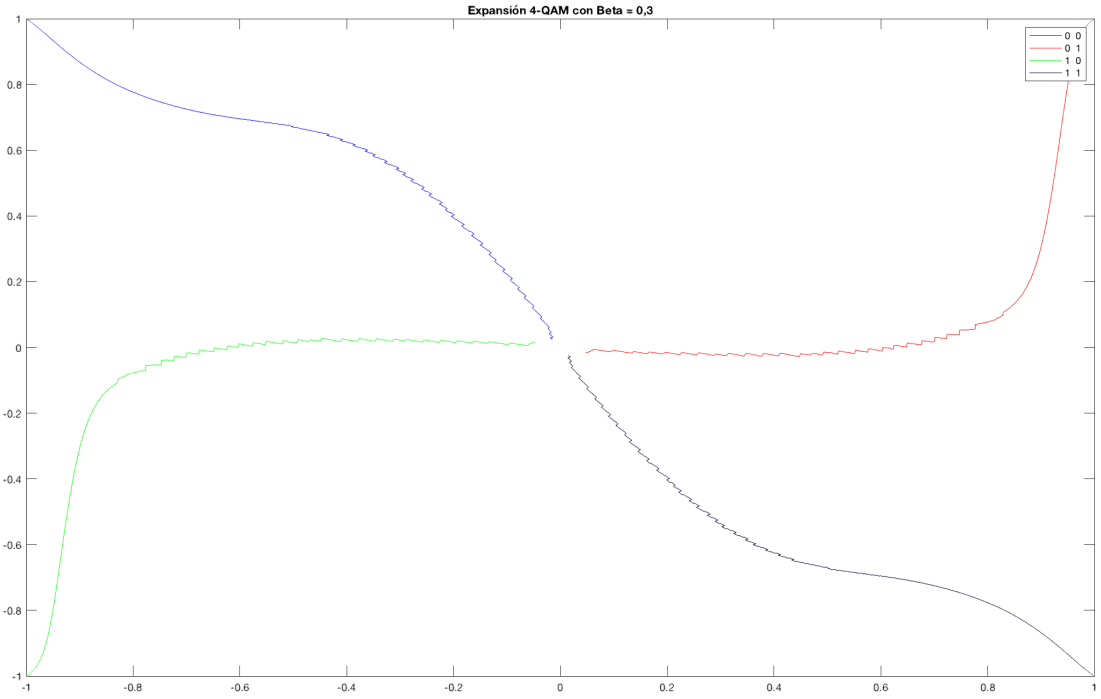

Fig. 5.1. Expansión constelación (Beta = 0,3)

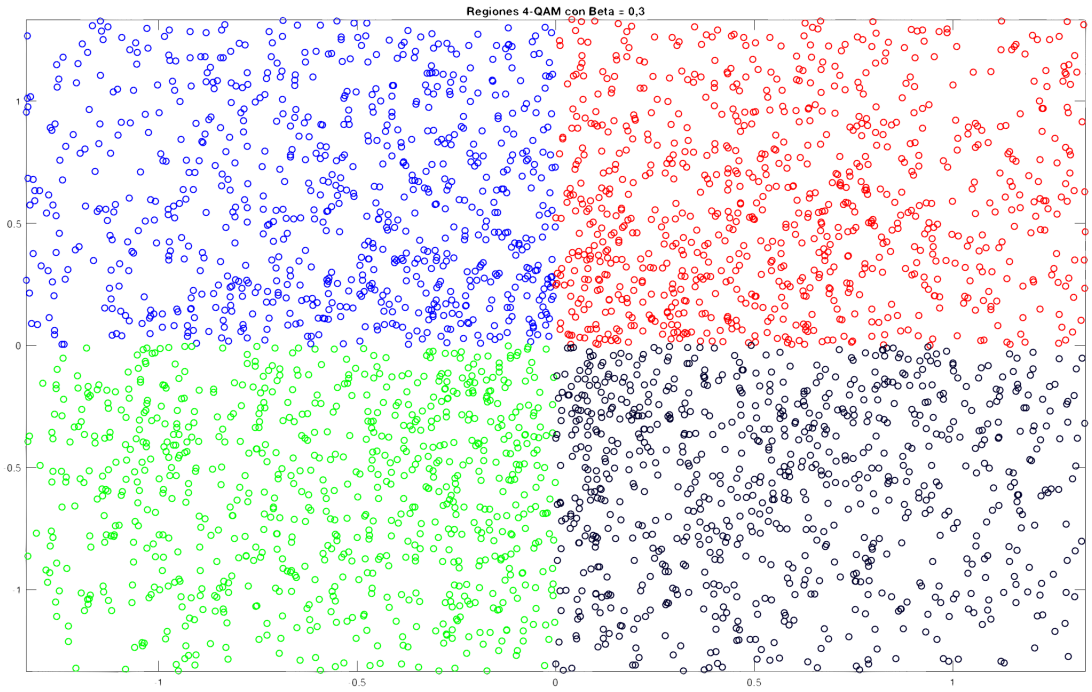

Fig. 5.2. Regiones de Decisión (Beta = 0,3)

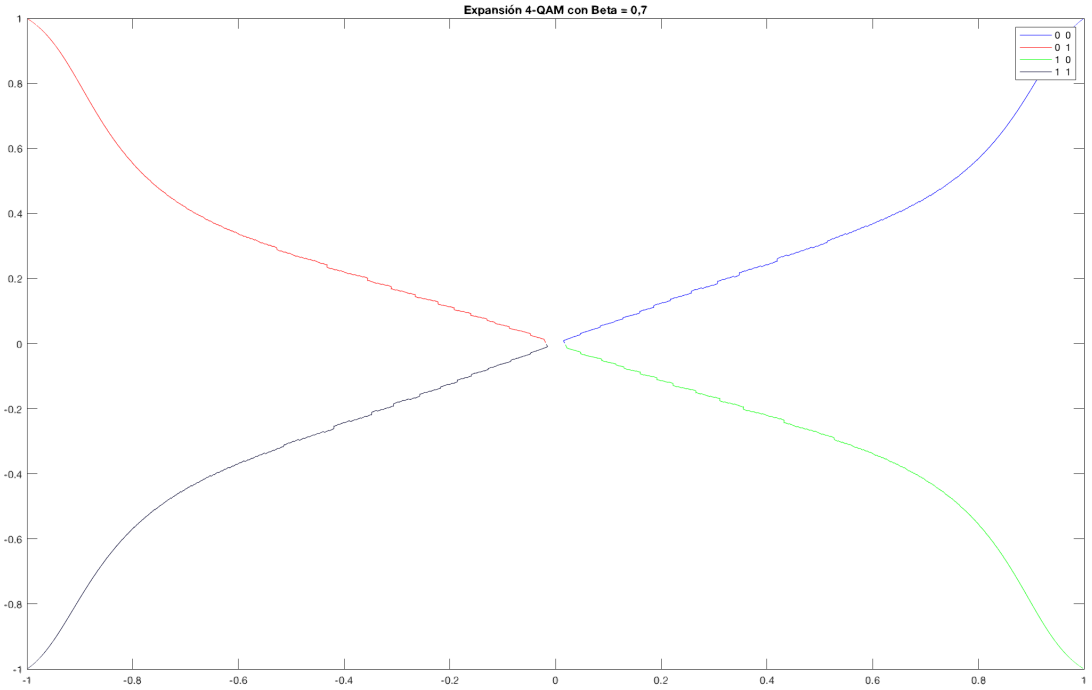

Fig. 5.3. Expansión constelación (Beta = 0,7)

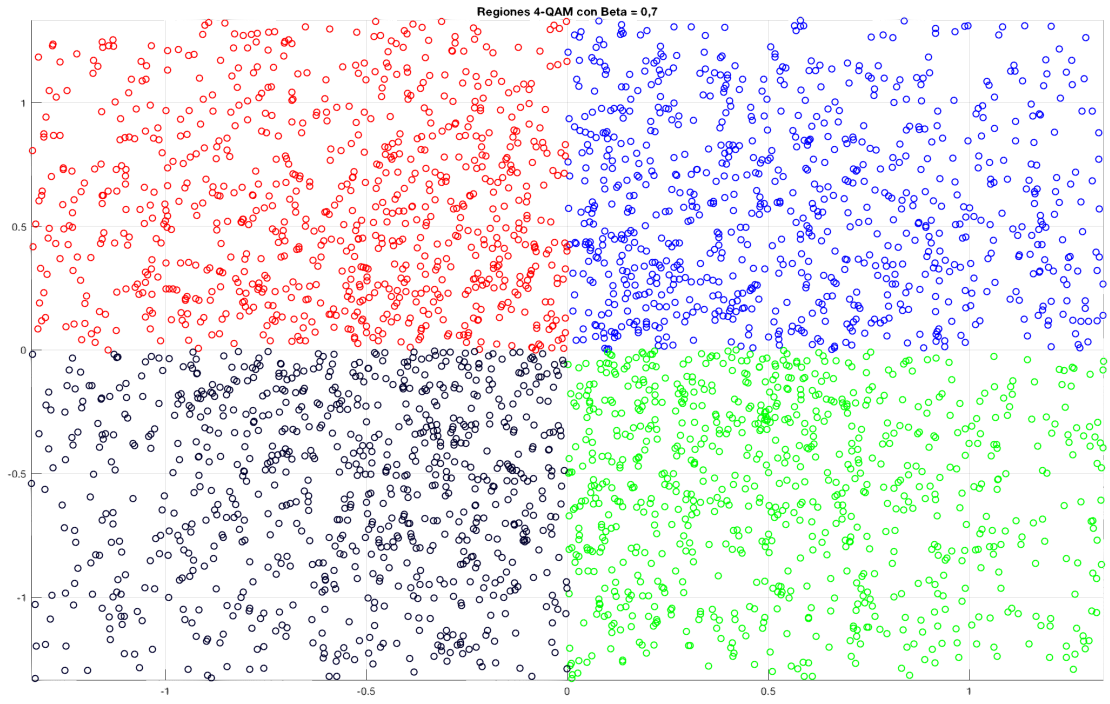

Fig. 5.4. Regiones de Decisión (Beta = 0,7)

Finalmente, en las figuras 5.5 y 5.6 se mostrarán los rendimientos para todos los Betas, en donde se aprecia que se consiguen siempre los mismos resultados, iguales al de la constelación 4-QAM normalizada.

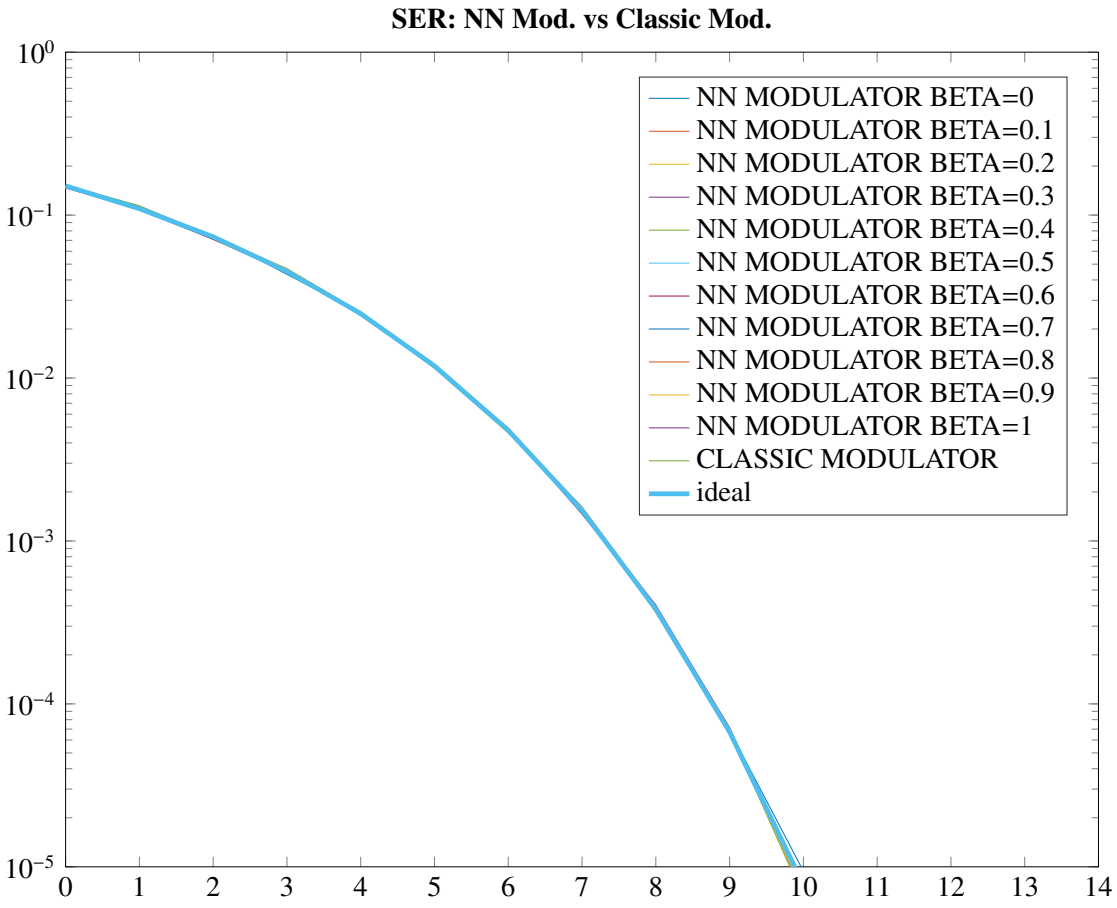

Fig. 5.5. SER 4-QAM propuesta original

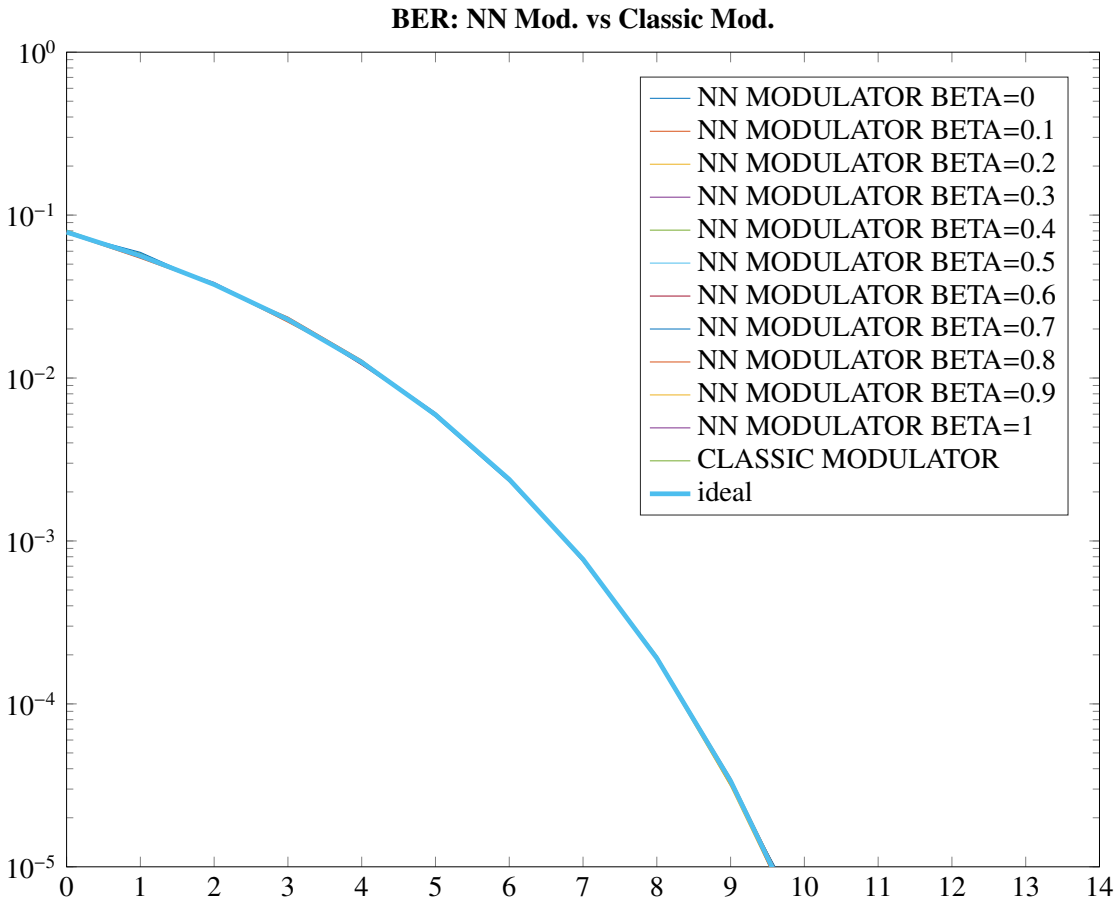

Fig. 5.6. BER 4-QAM propuesta original

Como se puede observar, las gráficas de SER y BER son prácticamente idénticas debido a la relación existente entre estas dos tasas de error.

$$SER = (log_2 M) \times BER \tag{5.1}$$

o lo que es lo mismo:

$$\frac{E_s}{N_0} = \frac{E_b}{N_0} \times (log_2 M) \tag{5.2}$$

Tras estos convincentes resultados iniciales, se decide saltar a la constelación relativa a este proyecto. El resultado esperado para esta ocasión es que según avance el sistema hacia un Beta próximo a 1, los rendimientos cada vez serán mejores hasta alcanzar en el máximo Beta un rendimiento similar al de las constelaciones 16-QAM, puesto que ese Beta implica el empleo del mismo criterio que el usado en el modelo estandarizado.

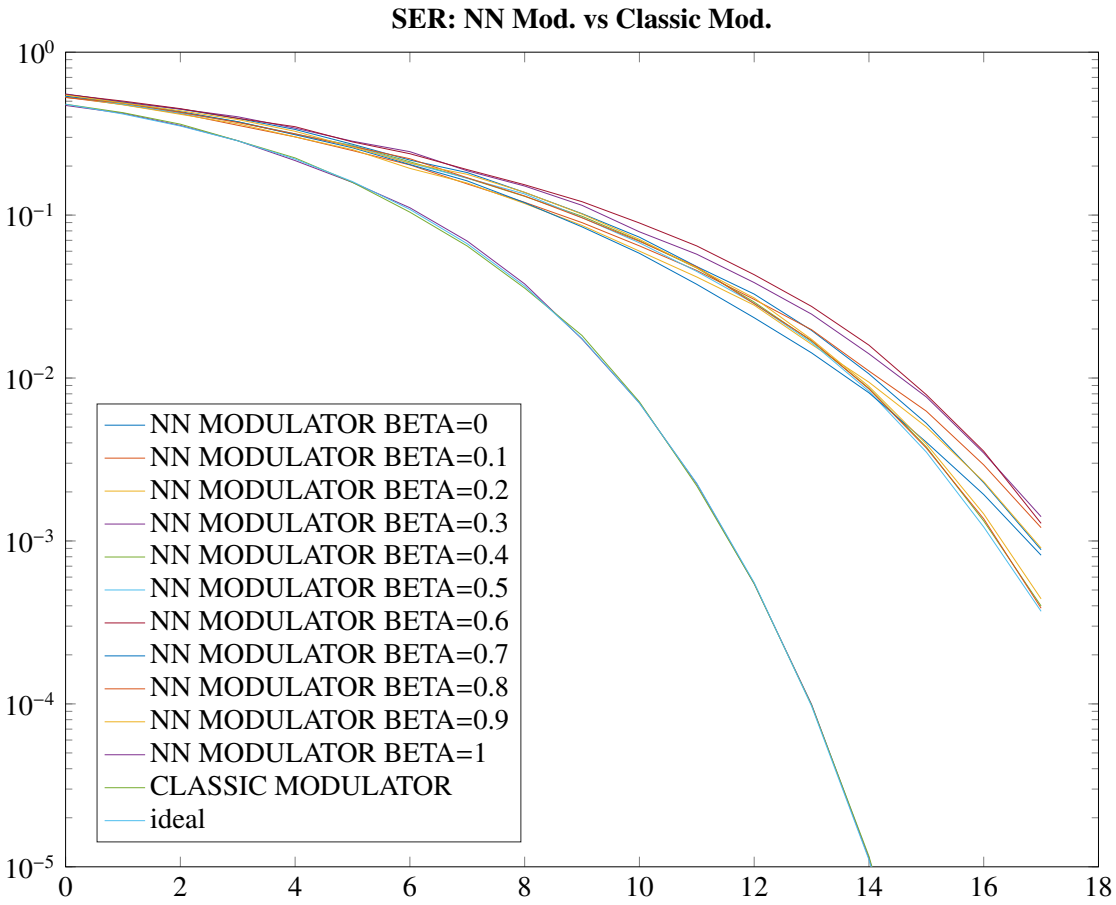

Fig. 5.7. SER 16-QAM propuesta original

Sin embargo, como se puede apreciar en la Fig.5.7, los resultados no son los esperados ya que los de la propuesta planteada quedan muy lejanos al rendimiento del modelo clásico. Este hecho supone que se prueben más trials para asegurarse que no se está próximo al objetivo del proyecto. Estas nuevas ejecuciones permitirán encontrar los puntos de fallo del sistema.

5.2. Problemas encontrados

Para encontrar estos problemas, se volverá sobre la constelación 4-QAM, debido a los menores tiempos de cómputo que ofrece. Tras la nueva ejecución de diversas simulaciones se encuentran dos resultados anómalos muy siginificativos.

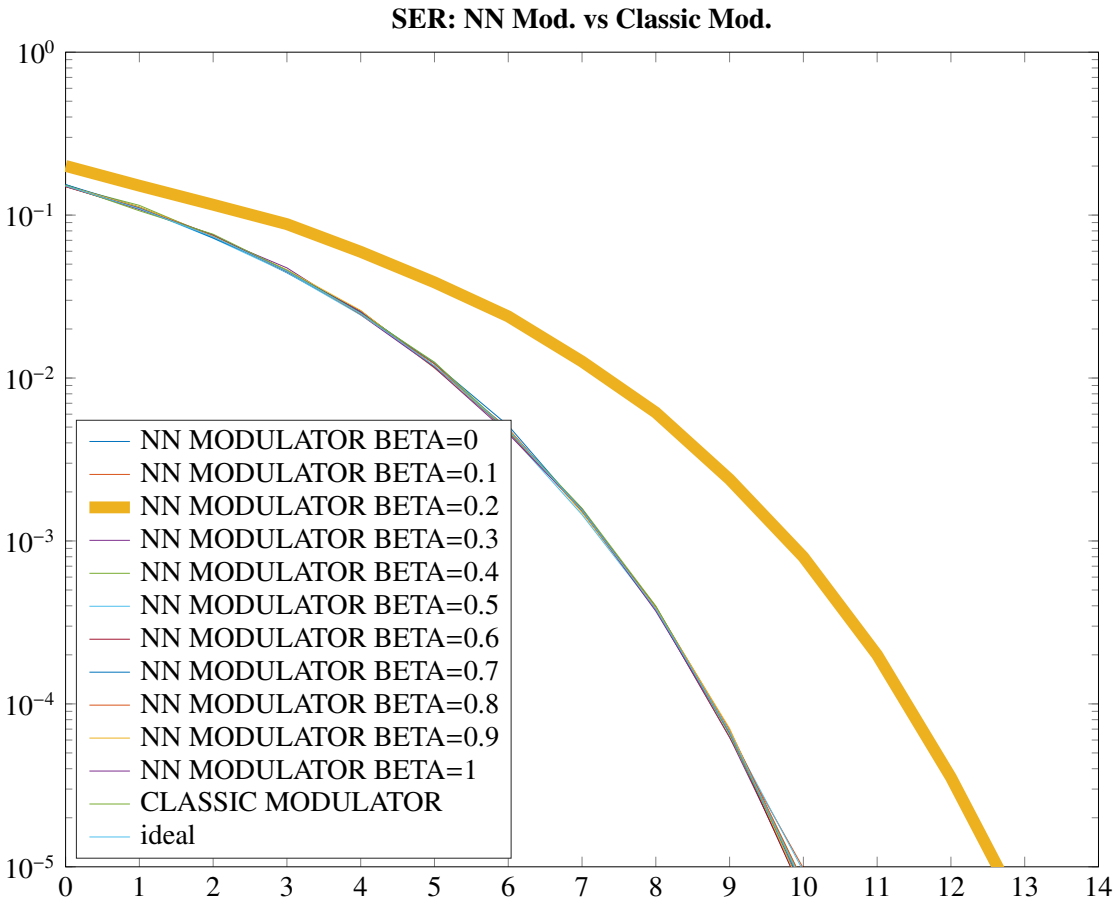

Fig. 5.8. SER 4-QAM primer resultado anómalo

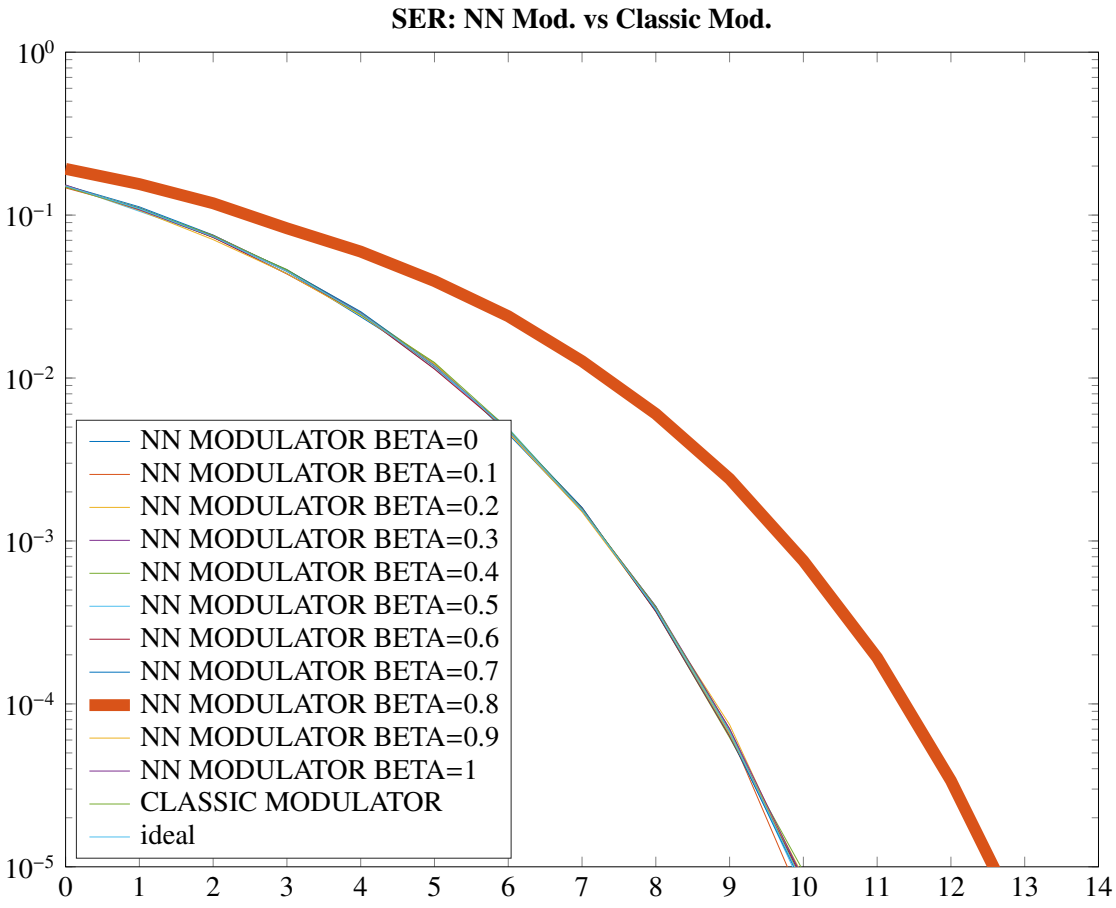

Fig. 5.9. SER 4-QAM segundo resultado anómalo

Como se puede observar en la Fig.5.8 y en la Fig.5.9, hay dos betas (0.2 y 0.8 respectivamente) que se salen del rango habitual de resultados. Esto hace pensar en que es probable que la red neuronal no esté convergiendo adecuadamente en esos casos, por lo que los resultados obtenidos no pueden acercarse a lo esperado. La razón a este hecho es que en esos determinados casos el sistema necesita un mayor número de iteraciones para converger, y en lugar de dárselas, la red neuronal termina su ejecución al llegar al número máximo, quedándose dichos puntos sin una maximización completa. Para ilustrarlo, se pueden ver las expansiones de ambos resultados anómalos. En ellas (Fig.5.10 y Fig.5.11) se comprueba que no han podido llegar a las esquinas de la gráfica, como si hacen el resto, siendo prueba inequívoca de la no convergencia del sistema y obteniendo de esta forma peores rendimientos.

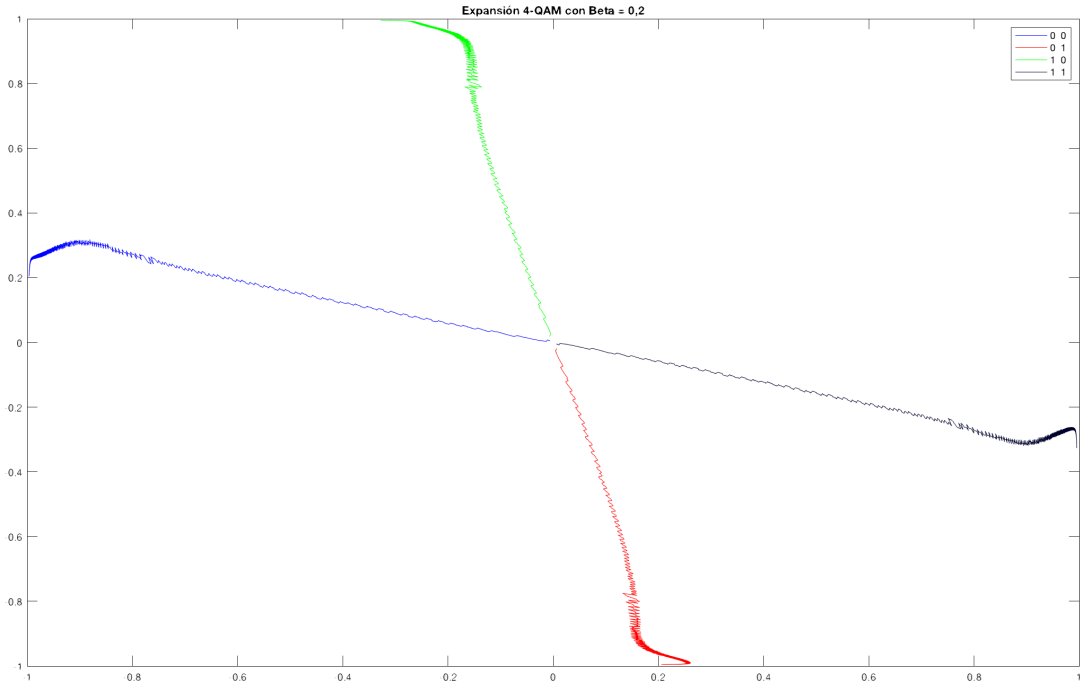

Fig. 5.10. Expansión constelación primer resultado anómalo (Beta = 0,2)

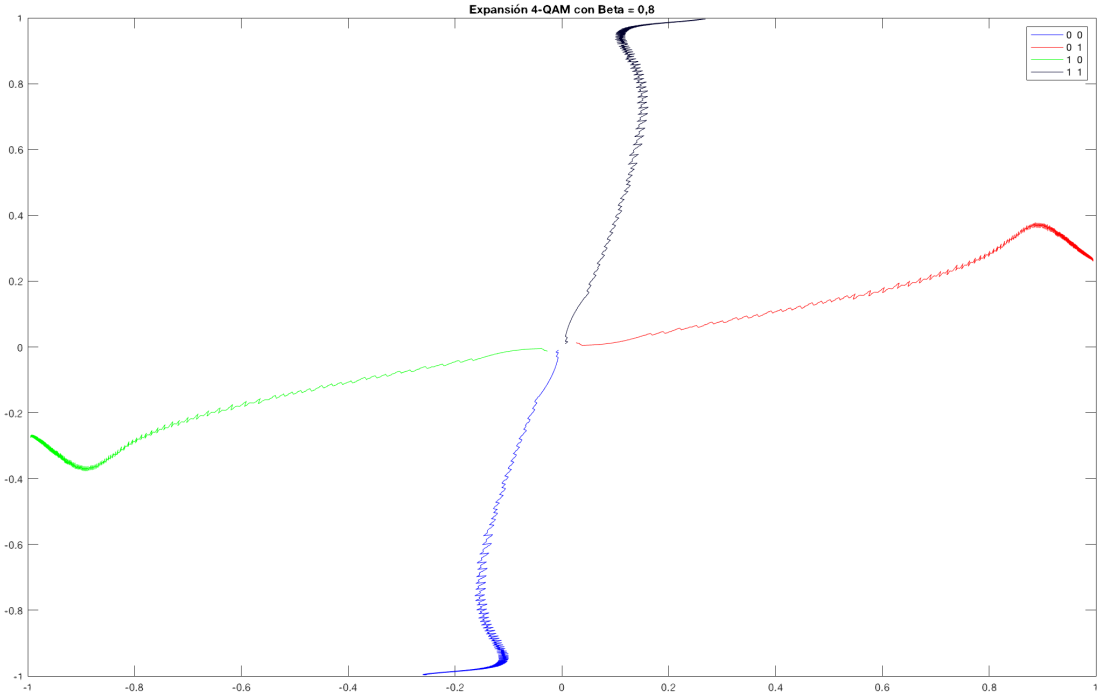

Fig. 5.11. Expansión constelación segundo resultado anómalo (Beta = 0,8)

Se puede intuir en ambas gráficas que los puntos están comenzando a dirigirse hacia las esquinas para su correcta maximización. Sin embargo, al llegar al número máximo de iteraciones el sistema corta la ejecución de forma abrupta quedándose los puntos en esas localizaciones.

Por tanto, se llega a la conclusión de que el sistema necesita en determinados casos un número mayor de iteraciones (respecto al máximo establecido en la sección 3.2) para lograr su convergencia. Por este motivo aparecen resultados anómalos en la constelación 4-QAM, y en la 16-QAM no se cumplen los objetivos fijados en el proyecto.

Como conclusión a este primer intento no satisfactorio del proyecto, se extrae que los resultados obtenidos de la red neuronal son impredecibles mientras que no se encuentre un algoritmo de parada adecuado. Debido a la aleatoriedad de la red neuronal, el número de iteraciones necesario en cada caso para llegar a la convergencia real fluctúa en cada ocasión, lo que hace necesario un número máximo de iteraciones variable en tiempo de ejecución.

Este hecho supone la adecuación de un nuevo criterio de parada de la red neuronal, y por tanto, una nueva propuesta para el proyecto.

5.3. Introducción a la nueva propuesta de mejora

Dada la variabilidad ya conocida en el número necesario de iteraciones para la convergencia de la red neuronal, se decide diseñar un nuevo algoritmo que sea lo más eficaz posible ante cualquier situación inesperada.

Por ello, se debe buscar un nuevo criterio para la condición de parada del sistema, permitiendo en algunos casos un número menor de iteraciones y en otros mayor, en función de las necesidades de convergencia.

En un primer momento se piensa en un algoritmo basado en la energía total de la constelación obtenida. De esta forma, cuanto más cercano a 1 sea ese valor (obtenido a partir de la varianza de la constelación), más expandida estará ésta. La idea es almacenar en una pila los tres valores más recientes de la energía de la constelación, calculados y almacenados cada 10 % de iteraciones del valor máximo establecido, y compararlos con el valor actual. De esta manera, si el valor actual es mayor que los anteriores y las iteraciones ya superan un 80 % del total, el número máximo de iteraciones permitidas cambiará, aumentando en un 1 %.

No obstante, este nuevo criterio se descarta al ser poco fiable, debido a las variaciones de energía durante la expansión (al no expandirse linealmente, sufre giros, vueltas, etc). Esto provoca que ante estos hechos la red neuronal determine que ya ha llegado a la convergencia final, mientras que en realidad el sistema no ha terminado de expandirse. Además, esta propuesta no emplea ninguna base teórica sobre la que se apoya el proyecto, por lo que se desarrolla una nueva idea para el algoritmo de parada.

Al ser el factor *loss* la función de coste que se trata de mejorar, se asume que también debe ser clave en el nuevo criterio de parada. Por tanto, en función de su historial de valores se incrementará, o no, el número máximo de iteraciones permitidas por el sistema de forma dinámica. En esta ocasión también se usa una pila con capacidad para tres valo-

res, así como la obtención del valor *loss* se hace de forma análoga al de la energía de la propuesta anterior (cada 10 % de iteraciones del número máximo establecido). Esta vez se decide que si dicho valor es mayor que dos de los tres valores almacenados en la pila, y ya se ha superado el 70 % de iteraciones máximas, este último incrementará un 2 % su valor. De esta forma, el número de iteraciones permitidas en el sistema variará de forma dinámica en función de las necesidades de la constelación utilizada, y será más flexible respecto al modelo descartado previamente.

6. PROPUESTA DE MEJORA

En este capítulo se tratarán las diversas modificaciones sobre el código tras la decisión de diseño de la nueva condición de parada, así como la implementación definitiva del modulador de la red neuronal.

6.1. Cambios respecto a la implementación de la Red Neuronal original

Respecto a la propuesta inicial de la red neuronal, se llevan a cabo diversas modificaciones. Sin embargo, la función *compute_graph_MODULATOR* y el resto que derivan de ella, permanecerán inalteradas, puesto que funcionaron correctamente desde el primer instante.

Los cambios que determinan esta nueva propuesta provienen de las modificaciones en la implementación del algoritmo de parada, la cual, en primera lugar, no tenía una lógica específica, sino más bien un carácter experimental.

```
1  with tf.Session(graph=my_graph) as session:
2              saver = tf.train.Saver()
3              tf.global_variables_initializer().run()
4              for it in range(num_max):
5                  fd={my_computed_dictionary['M']:messages,
                       my_computed_dictionary['K']:K,
                       my_computed_dictionary['Beta']:Beta}
6                  _,loss,minTrue,minPond,lcod,cod_tab,Ds,Dpon= session
                       .run([my_computed_dictionary['optimizer'],
                       my_computed_dictionary['loss'],
                       my_computed_dictionary['minTrue'],
                       my_computed_dictionary['minPond'],
                       my_computed_dictionary['var_cod'],
                       my_computed_dictionary['A'],
                       my_computed_dictionary['D_'],
                       my_computed_dictionary['D_2']], feed_dict=fd)
7                  for it2 in range(n):
8                      evolution[:,it2,it]=cod_tab[:,it2]
```

Como se puede apreciar en la línea 4 del código anterior, la condición de parada primitiva no ofrece ningún tipo de garantía y es el origen de los resultados anómalos en la constelación 4-QAM y la consecuencia de unos resultados fuera de lo esperado en la constelación 16-QAM.

Para solucionarlo, se define en primera instancia el número máximo de iteraciones inicial, cifrado en 10.000. A continuación, se crea la pila en donde se van introduciendo los valores *loss* calculados. La variable *it* indica la iteración por la que está atravesando

la red neuronal, mientras que *contador* se trata de una variable auxiliar para determinar el momento en el que se debe consultar el valor de *loss*. Por último, en el array *loss_vect* se van almacenando todos los valores de *loss* calculados para que puedan ser accesibles posteriormente.

```
        loss_vect=[]
        num_max=int(1e4)
        evolution=np.zeros([2**k,n,num_max], dtype=float)
        pila_loss=[0,0,0]
        with tf.Session(graph=my_graph) as session:
            saver = tf.train.Saver()
            tf.global_variables_initializer().run()
            it=0
            contador=0
```

6.2. Evolución final del Modulador de la Red Neuronal

Previamente a ser inicializada la sesión, se establece el *graph* que se empleará en la ejecución de la red neuronal. Esto no supone ningún cambio respecto a la propuesta original puesto que siempre se definió de esta manera.

```
[my_computed_dictionary,my_graph]=compute_graph_MODULATOR(batch_size
    ,k,n,cod_additional_density,cod_activation_add,alpha=alpha,lr=lr
    )
```

Tras establecerse, se inicia Beta a 0 con el fin de abarcar todo su rango, y gracias a los bucles establecidos, se recorren todos los Betas y se obtiene el número de trials que se guste. A continuación, se definen las variables necesarias y se inicia la sesión, tal y como se mostraba previamente en el fragmento de código 6.1.

Tras estas consideraciones previas, se pasa a introducir las comprobaciones propias del nuevo algoritmo de parada, así como su escapatoria para ejecuciones que requieran un número ingente de iteraciones para converger.

```
while (it<num_max and it<1e5):
```

Si se cumple dicha condición, la sesión se ejecutará permitiendo la obtención de los valores resultantes de las instrucciones del graph, así como los puntos por donde la constelación se ha expandido. Además, la variable *contador*, se incrementará en cada ejecución una unidad, ya que se empleará en otra de las condiciones posteriores del algoritmo de parada.

```
fd={my_computed_dictionary['M']:messages,my_computed_dictionary['K'
    ]:K,my_computed_dictionary['Beta']:Beta} _,loss,minTrue,minPond,
    lcod,cod_tab,Ds,Dpon= session.run([my_computed_dictionary['
```

```
    optimizer'],my_computed_dictionary['loss'],
    my_computed_dictionary['minTrue'],my_computed_dictionary['
    minPond'],my_computed_dictionary['var_cod'],
    my_computed_dictionary['A'],my_computed_dictionary['D_'],
    my_computed_dictionary['D_2']], feed_dict=fd)

      for it2 in range(n):
         evolution[:,it2,it]=cod_tab[:,it2]
      constel = evolution[:,:,it]
      contador+= 1
```

A continuación, se establece una nueva condición, la cual es indispensable superar para poder seguir atravesando el sistema de parada.

```
      ##cuando llegamos a un 10% de num_max
      if contador==int(num_max*0.01):
```

Si no fuera así, simplemente se incrementaría una unidad el número de iteraciones de la red neuronal y se volvería a proceder a su ejecución en caso de cumplir con la condición 6.2. Si por el contrario el sistema converge antes de llegar al 10 %, éste seguiría ejecutándo iteraciones hasta llegar a ese valor, puesto que este nuevo algoritmo también está basado en cierta medida en el número de iteraciones. No obstante, tras todas las simulaciones realizadas nunca se ha dado este caso.

Tal y como se explicó en la sección 5.3, todos los cálculos y obtención de valores relativos y necesarios para la nueva condición de parada se realizarán cada 10 % del número de iteraciones máximo actual. De esta forma, se le da cierto margen al sistema para evolucionar, comprobando su expansión pasados este número de iteraciones, evitando así realizar cálculos constantemente.

Por ello, suponiendo que se atraviesa este nuevo filtro al haber llegado a esta cifra adaptativa de iteraciones, se pasa a calcular cuántos valores de la pila cumplen que la diferencia entre el valor de *loss* obtenido y los guardados en la pila (*pila_loss*), son mayores que 10^{-4} . De esta manera, se previene de que la red neuronal no converja nunca debido a que los nuevos valores de *loss* calculados siempre sean mayores que sus antecesores por decimales insignificantes.

```
         tmp_loss=sum((loss-pila_loss)>1e-4)
```

Por tanto, gracias a *tmp_loss* se puede evaluar la segunda y última nueva condición de parada del sistema. Esta nueva condición, tal y como se detalló en la sección 5.3, supone para su aprobación que el nuevo *loss* sea mayor que dos de los valores previos almacenados en la pila (de tamaño 3), así como que el número de iteraciones llevadas a cabo supere el 70 % del número máximo actual.

```
         if (tmp_loss>=2 and it>=0.7*num_max):
```

Al no tener que ser mayor *loss* que todos los valores de la pila, se le otorga al sistema cierto grado de flexibilidad frente a posibles inicios de convergencia inadecuados, permitiendo así la posibilidad de corregir el rumbo de la expansión.

Dentro de esta condición se llevan a cabo las labores de incremento del número máximo de iteraciones permitido, a la vez que se redimensiona el array sobre el que se guardan los datos que se obtienen en las iteraciones posteriores.

```
                inc=int(num_max*0.01)
                num_max+=inc
                evolution=np.concatenate((evolution,np.empty([2**k,n,inc
                    ], dtype=float)),axis=2)
```

Finalmente, y aunque esta última condición planteada no haya sido satisfactoria, se inserta el nuevo valor en la *pila_loss* (y por tanto el último valor debe ser eliminado para dejar hueco), además de guardarse en el vector *loss_vect* y establecer de nuevo el contador a cero.

7. RESULTADOS Y CONCLUSIONES DEL PROYECTO

En este capítulo se expondrán los diferentes resultados finales del trabajo, así como sus comparaciones frente al sistema tradicional. Finalmente, se señalarán las conclusiones que se obtienen tras el análisis previo.

7.1. Análisis de las gráficas

Tras la ejecución de la red neuronal se trasladan los datos resultantes, como tras la implementación original, a Matlab con el objetivo de conseguir las diferentes representaciones gráficas que se requieren para lograr una conclusión razonada del proyecto. En esta ocasión se llevan a cabo cuatro simulaciones para cada Beta con el objetivo de determinar que los resultados son correctos y fiables en cualquier ocasión.

En primer lugar se comprueban los resultados de SER para todos los Betas en cada simulación. Como se puede apreciar en cada ejecución, los Betas más próximos a cero otorgan un peor rendimiento al sistema, mientras que según se aumenta su valor y éste se acerca a uno, su rendimiento mejora considerablemente hasta lograr finalmente conseguir una eficiencia igual a la ideal y al modelo tradicional.

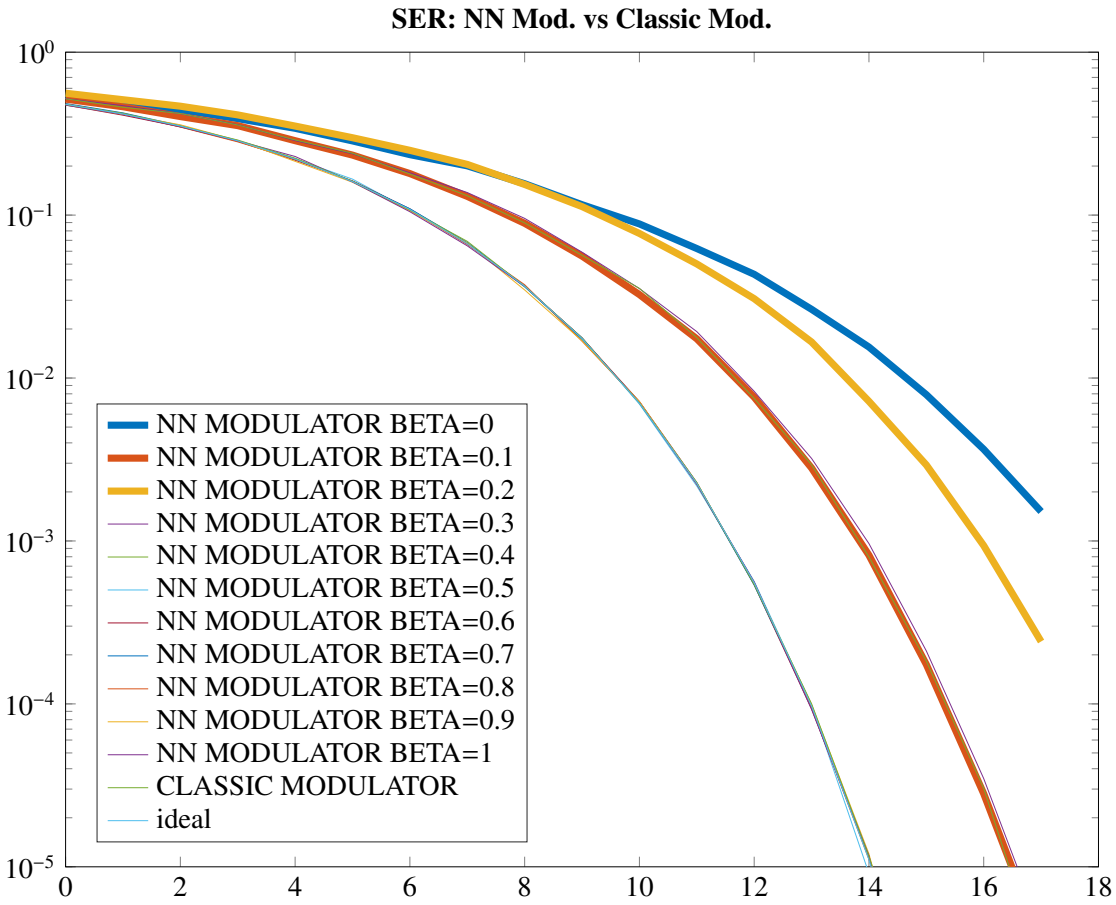

Fig. 7.1. SER 16-QAM nueva condición de parada (Trial 0), remarcados Betas bajas

En este caso, se puede apreciar en la figura 7.1, en la cual se resaltan los Betas más pequeños, que éstos proporcionan un rendimiento inferior a los altos. Sin embargo, no se trata de una mejora lineal, puesto que en este trial el rendimiento de $Beta = 0,1$ es superior al de $Beta = 0,2$.

Para este ejemplo, se aprecian claramente los factores que infieren en el pobre rendimiento ofrecido de estos Betas. Para determinar el motivo, se recurre a comparar la regiones de decisión ideal específicas de este trial, con las regiones de estas Betas.

Comenzando por el de peor rendimiento (*Beta=0*), sus regiones de decisión, son claramente distintas que las ideales. El motivo es la deficiente expansión de dicha contelación.

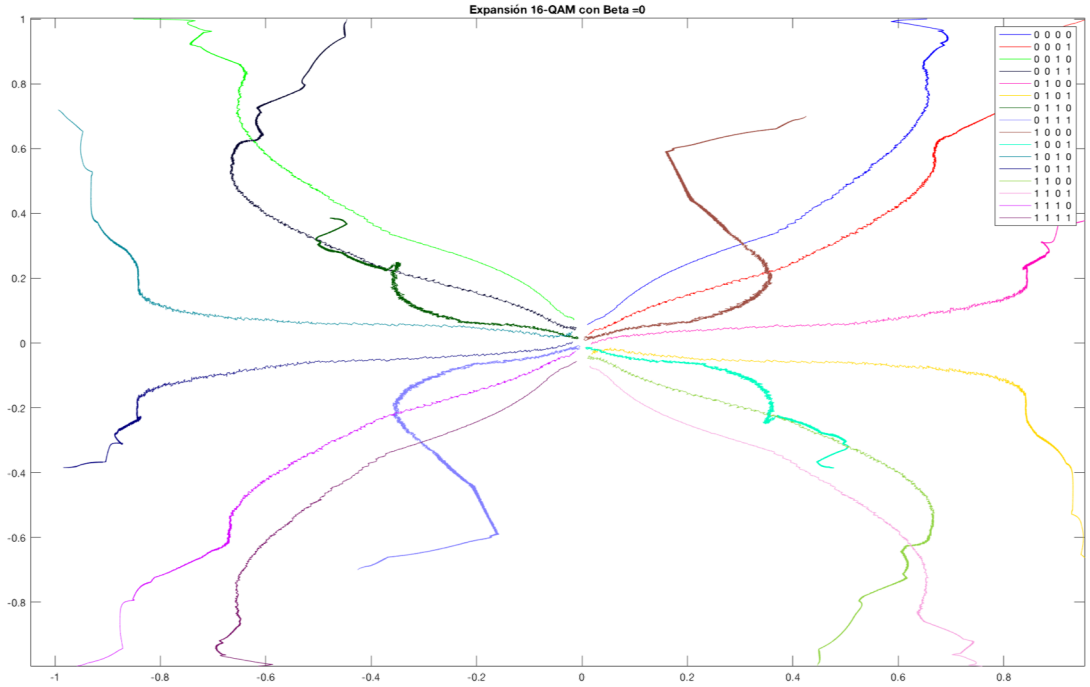

Fig. 7.2. Expansión constelación Beta=0 (Trial 0)

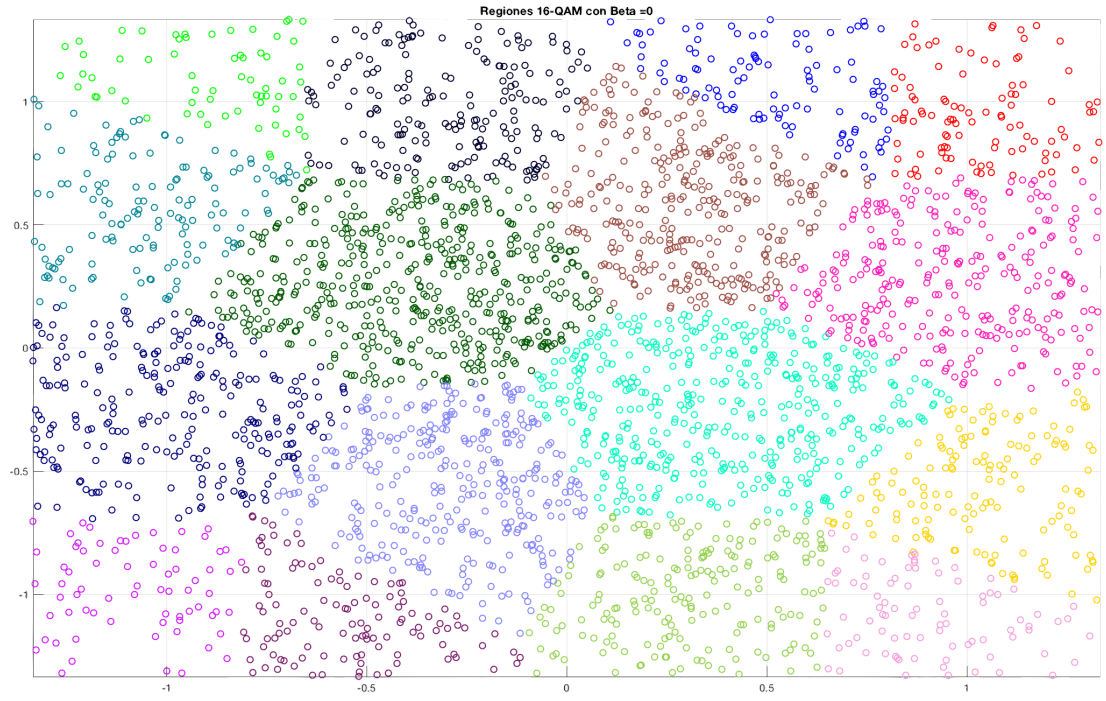

Fig. 7.3. Región de Decisión Beta=0 (Trial 0)

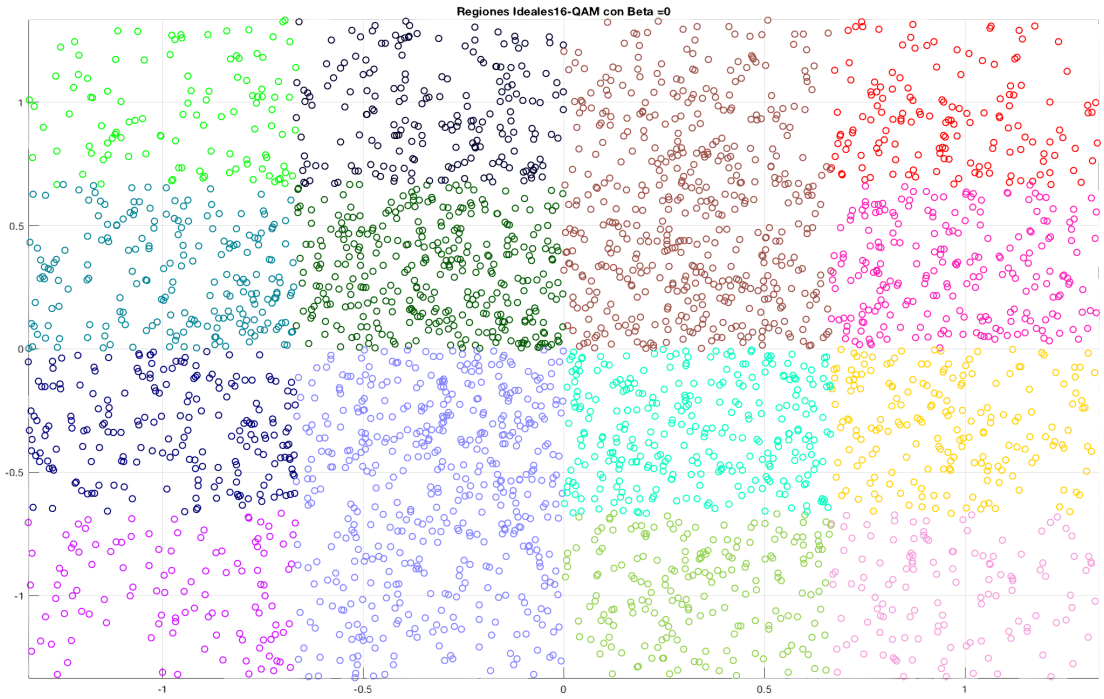

Fig. 7.4. Región de Decisión Ideal Beta=0 (Trial 0)

En el caso ideal, mostrado en la figura 7.4, se aprecia el clásico estilo de rejilla habitual de los esquemas de modulación, mientras que la figura 7.3 muestra unas regiones completamente amorfas, sin orden en la división aparente.

No obstante, a medida que el valor de Beta va incrementando, en general, el rendimiento del sistema mejora frente a esos Betas bajos, puesto que cada vez el modelo propuesto se parece más al tradicional. Por ejemplo, analizando los casos de los Betas resaltados en la figura 7.5, muestran una mejoría en el rendimiento frente a las figuras expuestas anteriormente.

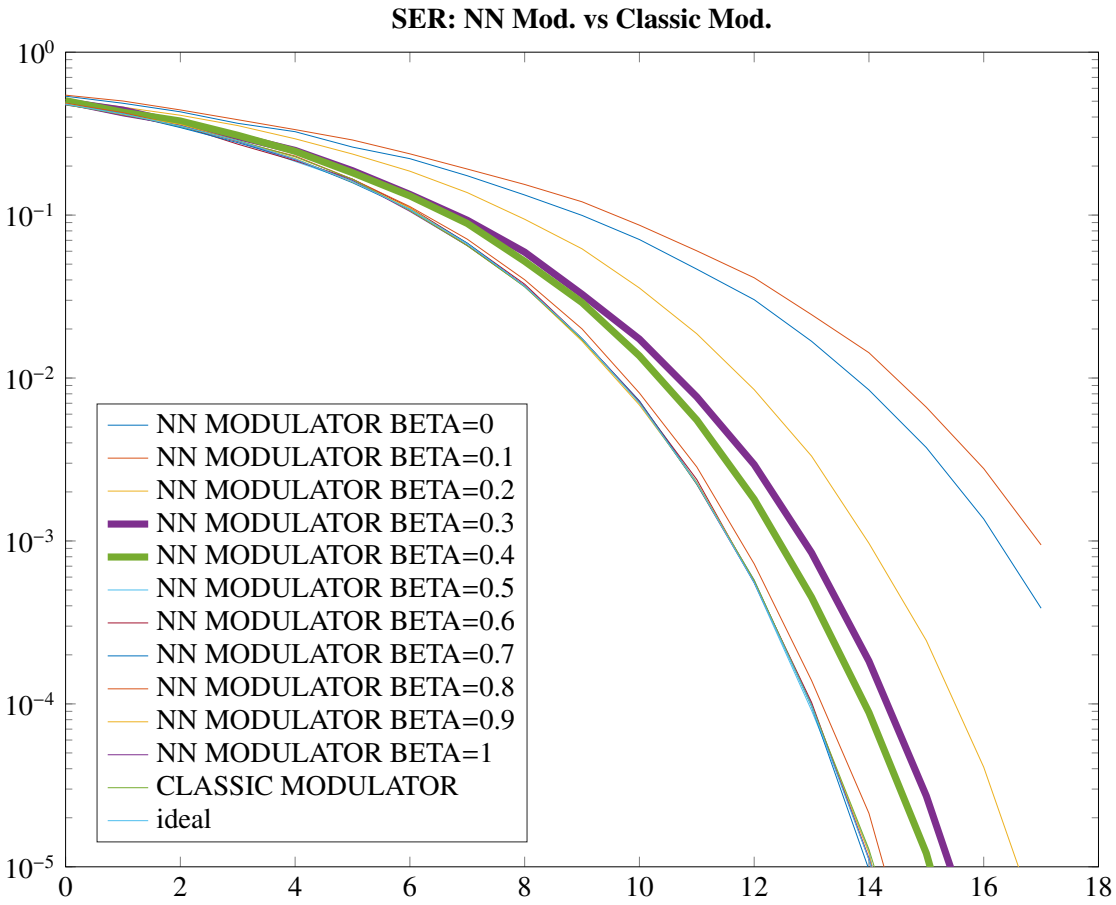

Fig. 7.5. SER 16-QAM nueva condición de parada (Trial 1)

Por este motivo, en sus regiones de decisión, se aprecian mayores similitudes frente a sus análogas ideales (Fig.7.8 y Fig.7.11), respecto a las figuras 7.3 y 7.4.

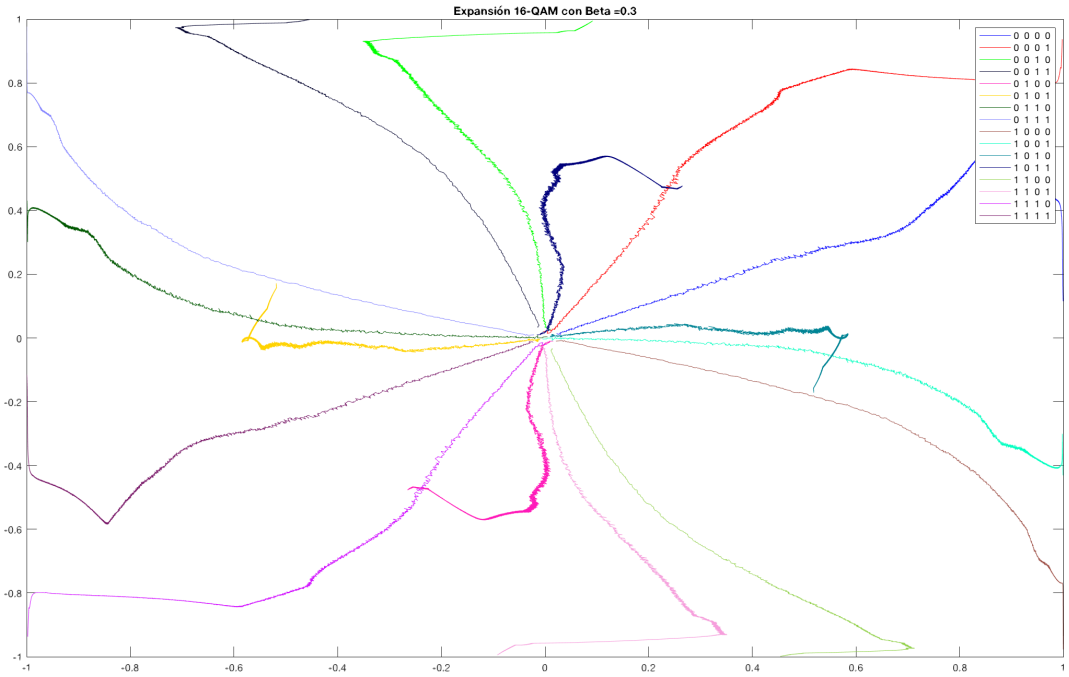

Fig. 7.6. Expansión constelación Beta=0,3 (Trial 1)

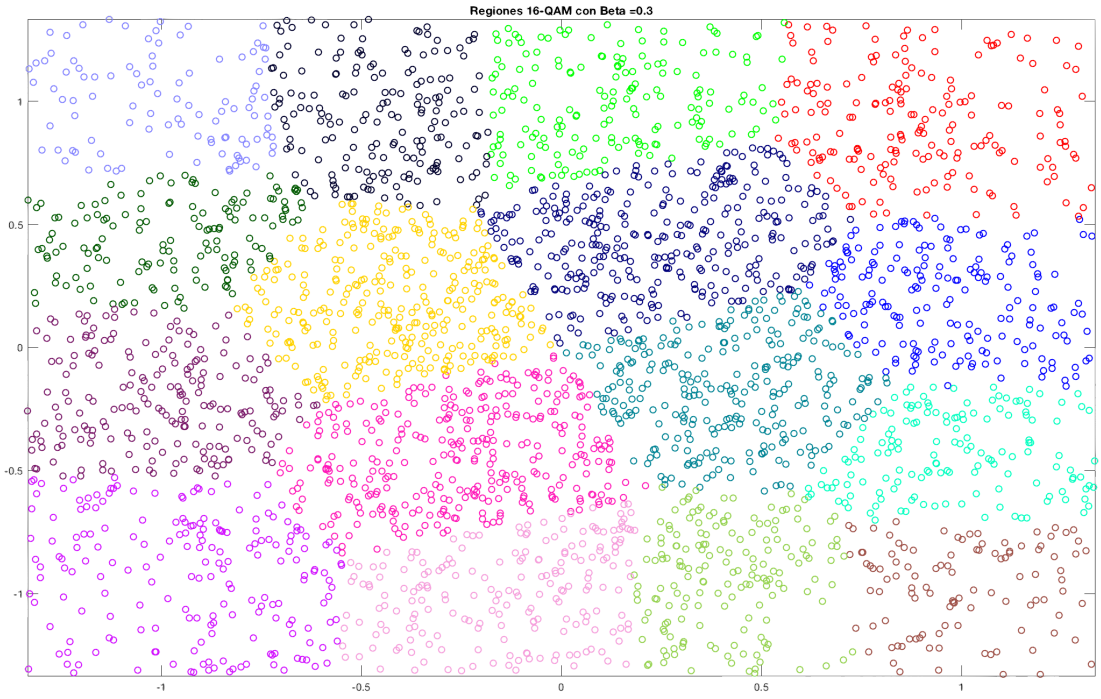

Fig. 7.7. Región de Decisión Beta=0,3 (Trial 1)

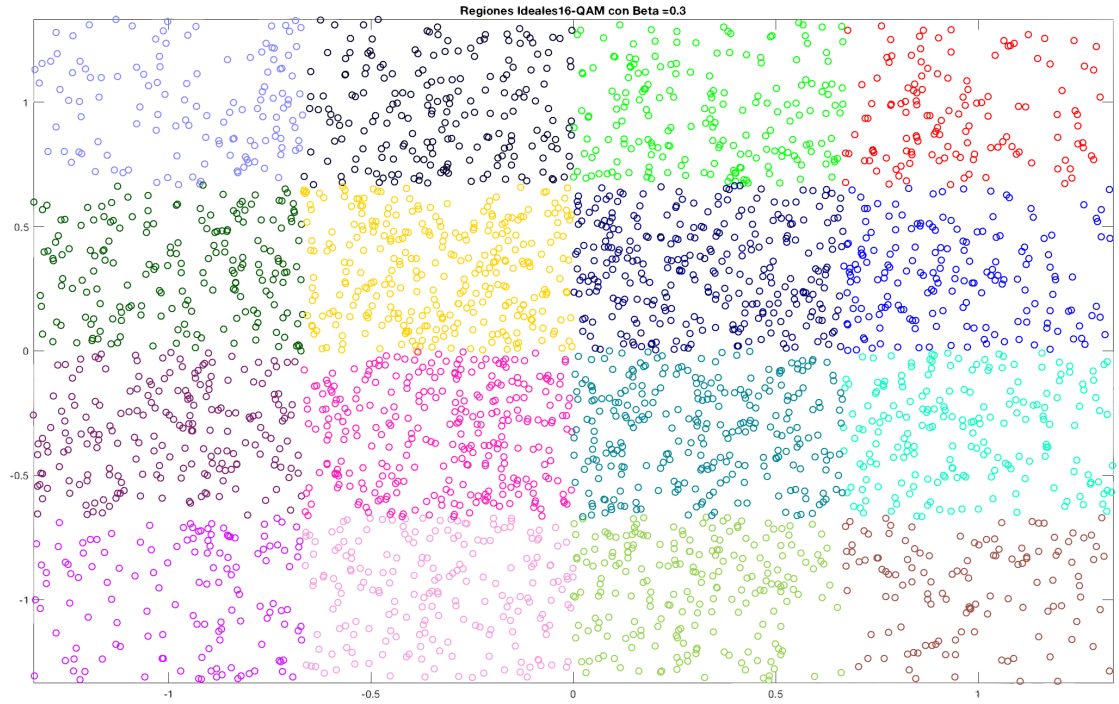

Fig. 7.8. Región de Decisión Ideal Beta=0,3 (Trial 1)

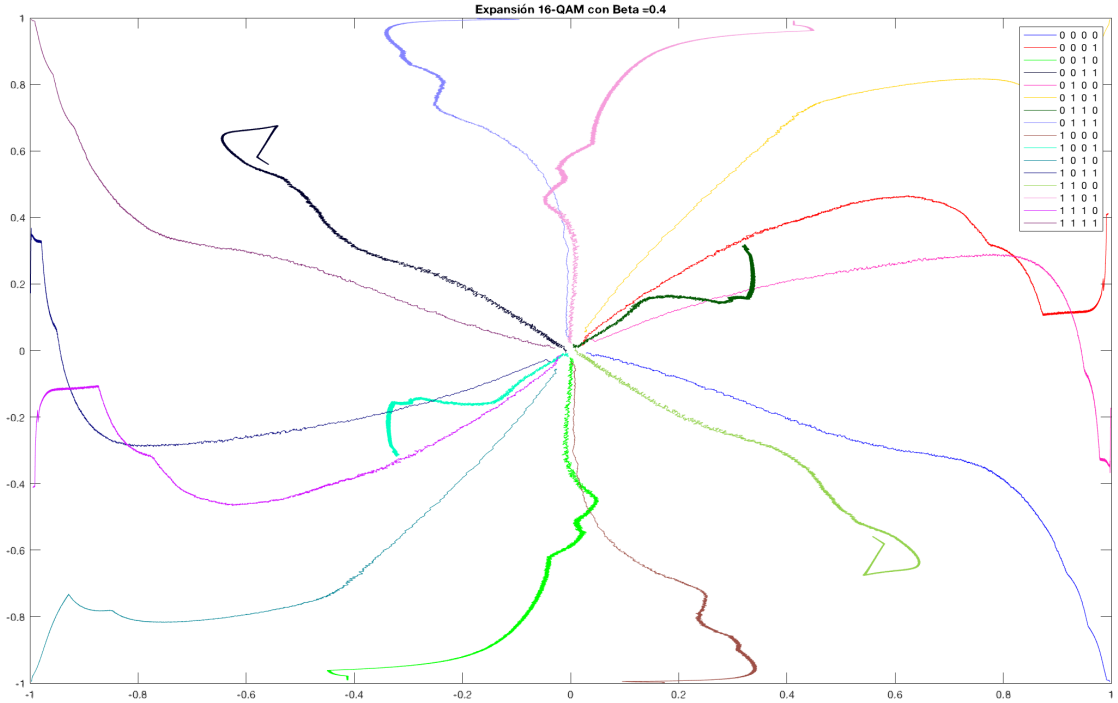

Fig. 7.9. Expansión constelación Beta=0,4 (Trial 1)

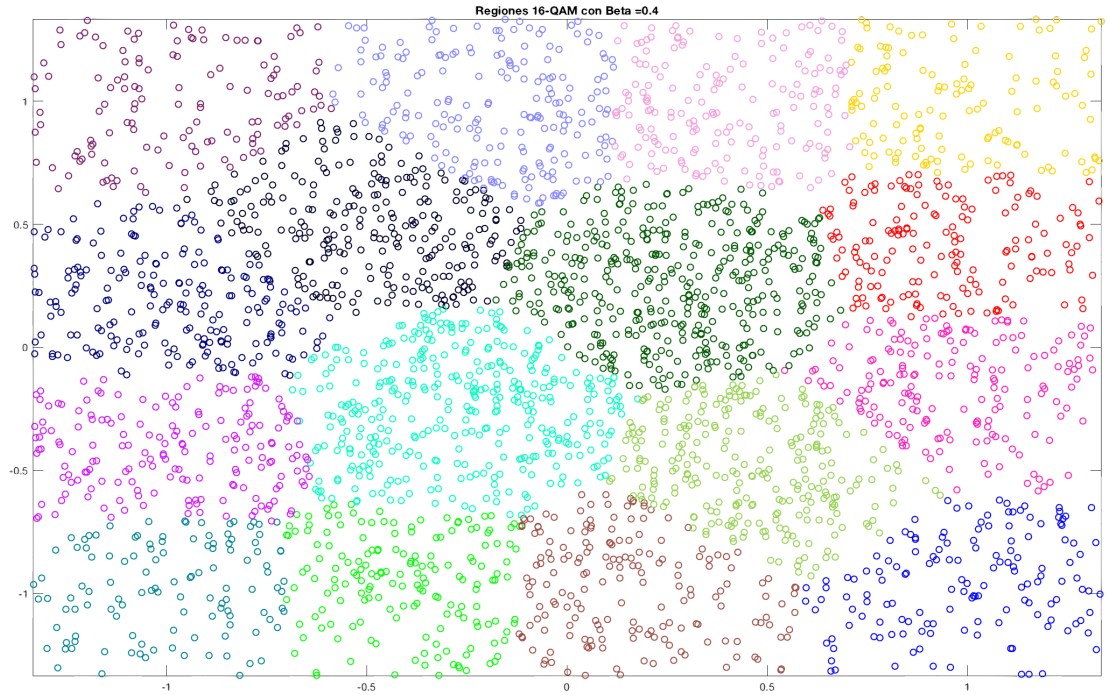

Fig. 7.10. Región de Decisión Beta=0,4 (Trial 1)

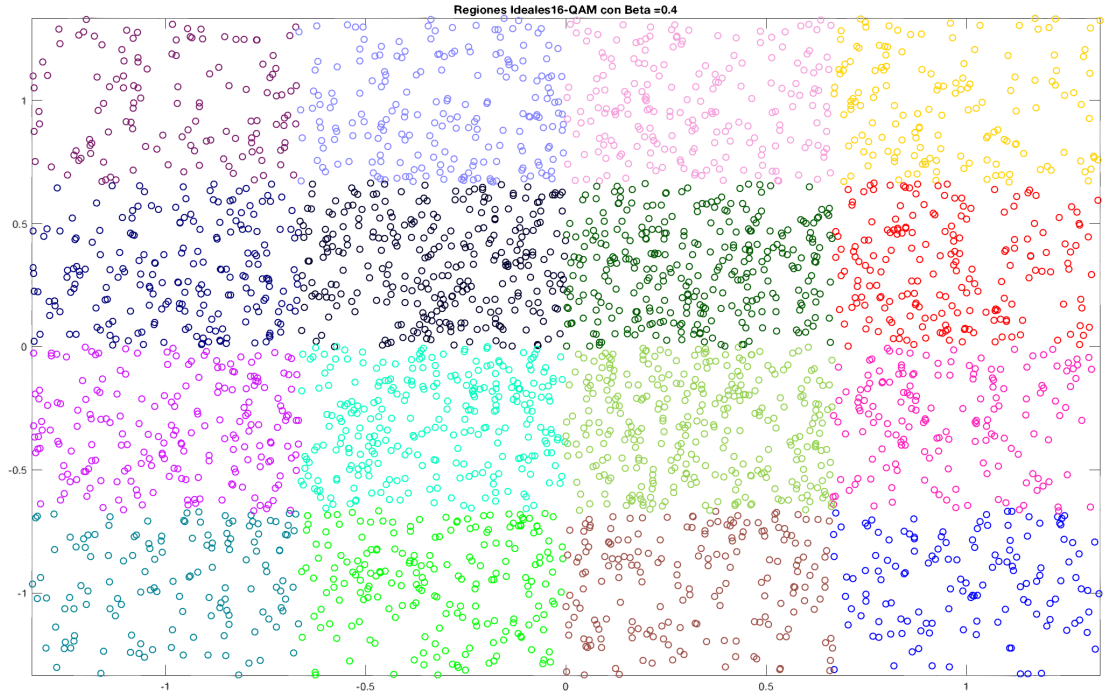

Fig. 7.11. Región de Decisión Ideal Beta=0,4 (Trial 1)

Este hecho supone que los puntos de la constelación se van expandiendo de tal forma que ya se aproximan ligereramente a la formación tradicional, motivo por el cual el rendimiento se incrementa frente a los Betas anteriores.

Aumentando el valor de Beta, y centrándose en el siguiente rango de valores (0.5, 0.6 y 0.7), se muestra a continuación sus tasas de error. Como se aprecia en la figura 7.12, estos Betas remarcados consiguen unos rendimientos iguales que en el caso ideal y que en el modelo tradicional. No obstante, para poder afirmar este hecho de forma categórica y no anecdótica, se deben ponderar todos sus datos recopilados, con el fin de obtener una media de todos sus resultados.(Fig.7.18)

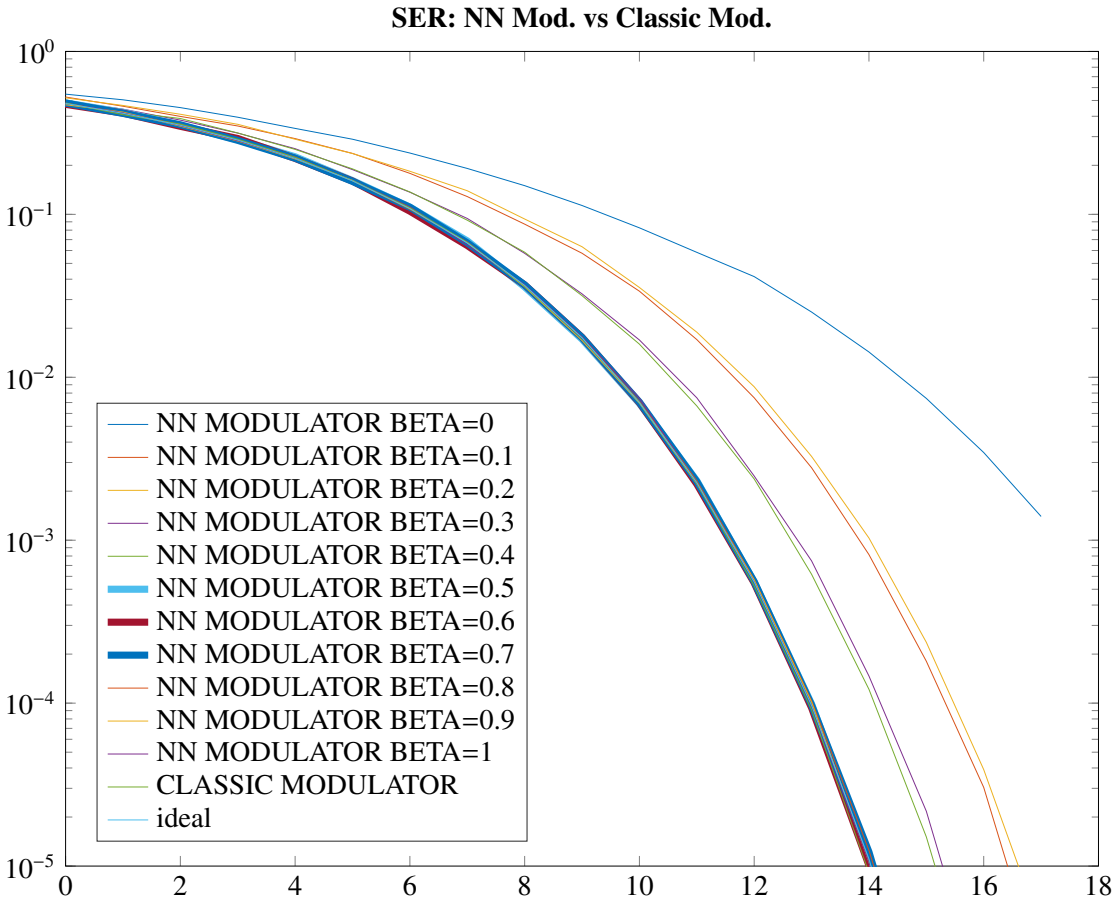

Fig. 7.12. SER 16-QAM nueva condición de parada (Trial 2)

Para los Betas señalados en la Fig.7.12, al ser idénticos y tener un rendimiento similar al tradicional, simplemente se mostrará la expansión y las regiones de decisión de *Beta=0,6*.

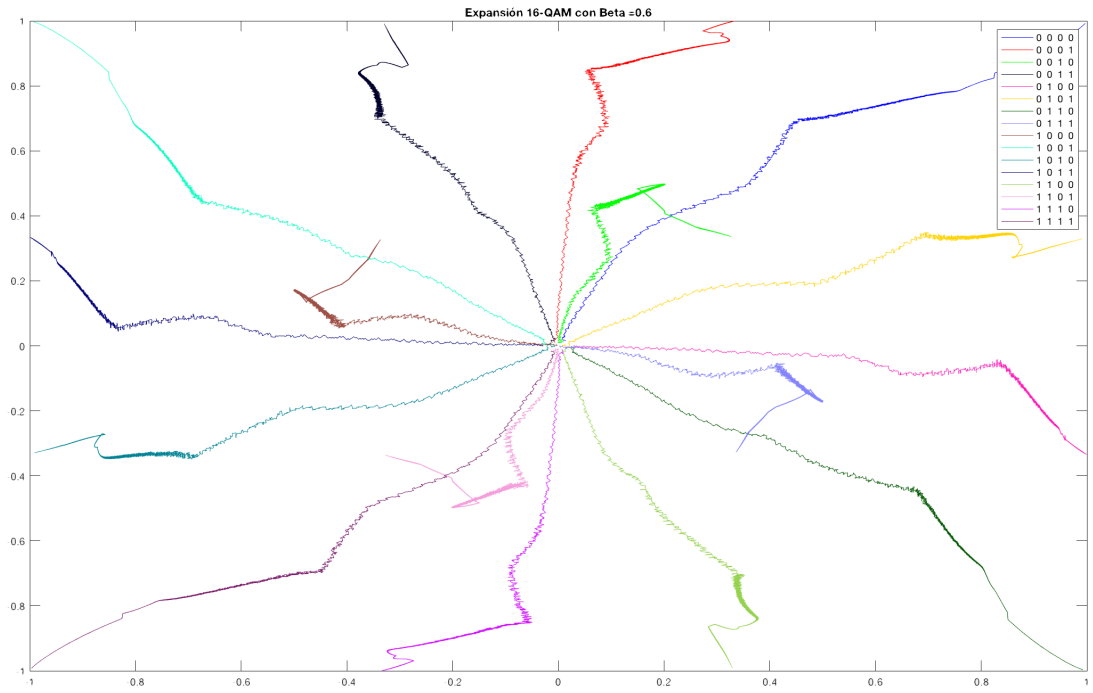

Fig. 7.13. Expansión constelación Beta=0,6 (Trial 2)

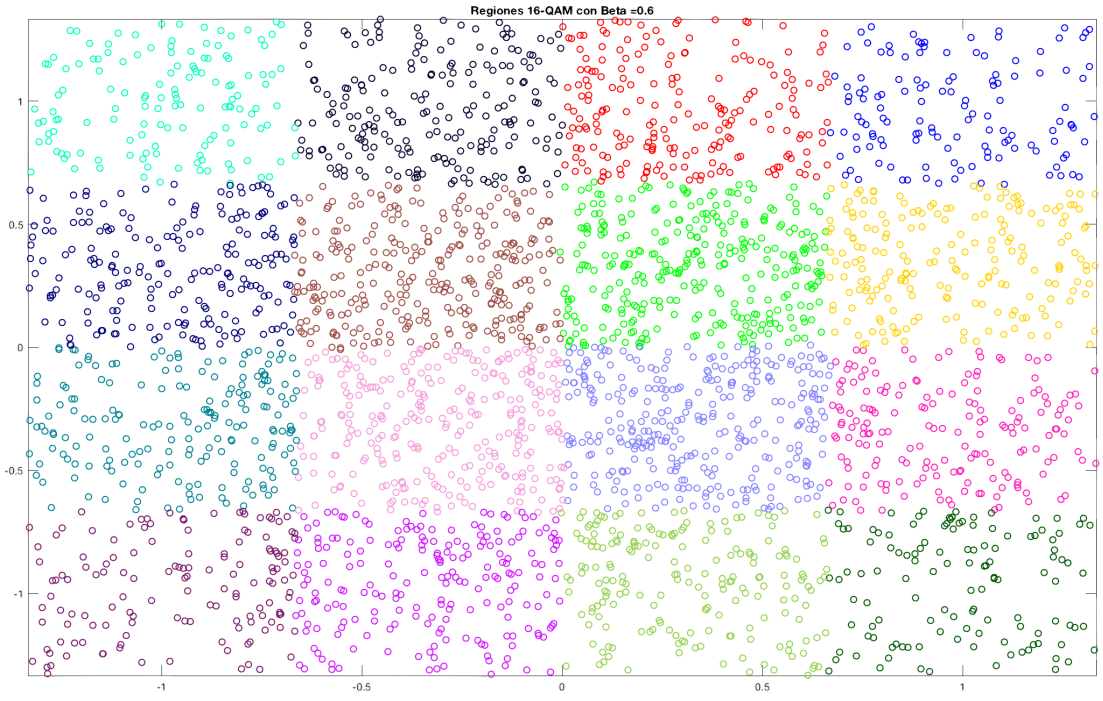

Fig. 7.14. Región de Decisión Beta=0,6 (Trial 2)

Para concluir este análisis individualizado por secciones de valores de Beta, llegamos al estudio de sus últimos valores posibles. En esta ocasión sucede algo similar a lo ocurrido en la Fig.7.12. Para ejemplificarlo, se emplearán las gráficas resultantes de la ejecución

para *Beta=0,9*.

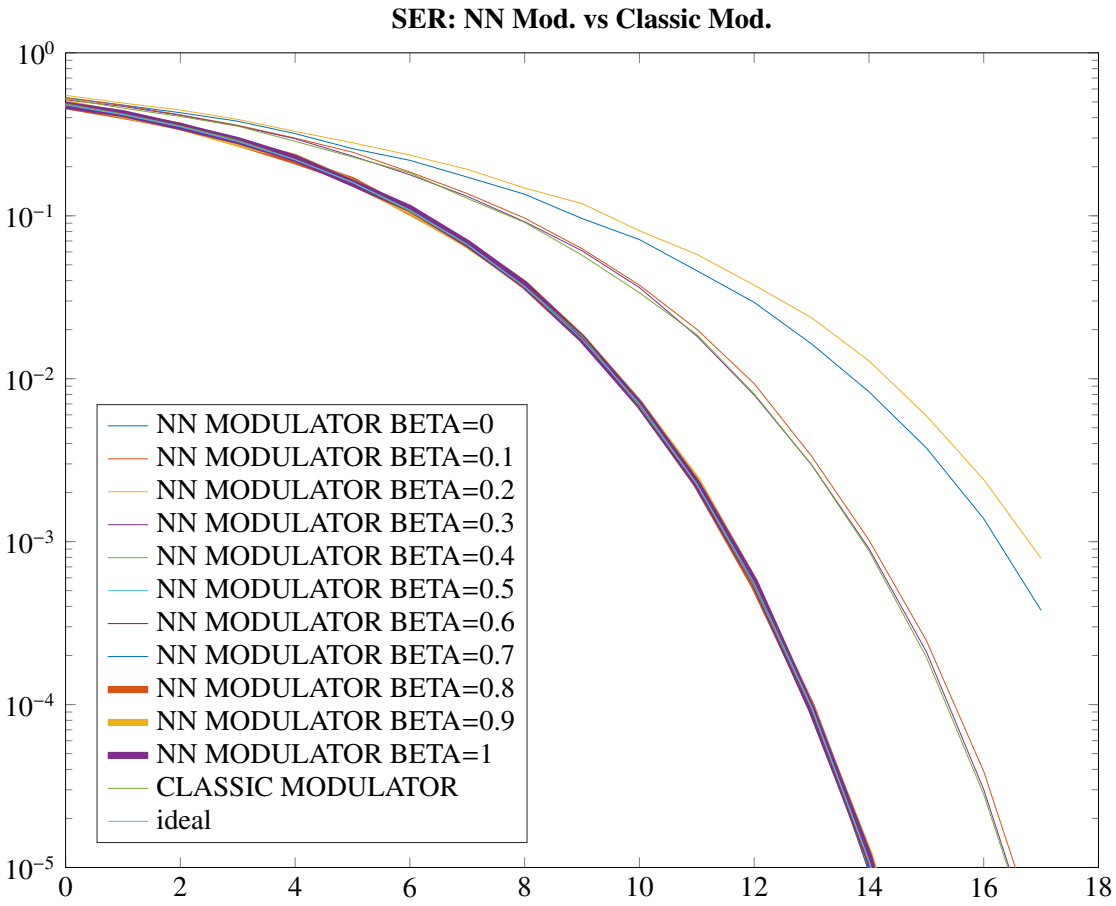

Fig. 7.15. SER 16-QAM nueva condición de parada (Trial 3)

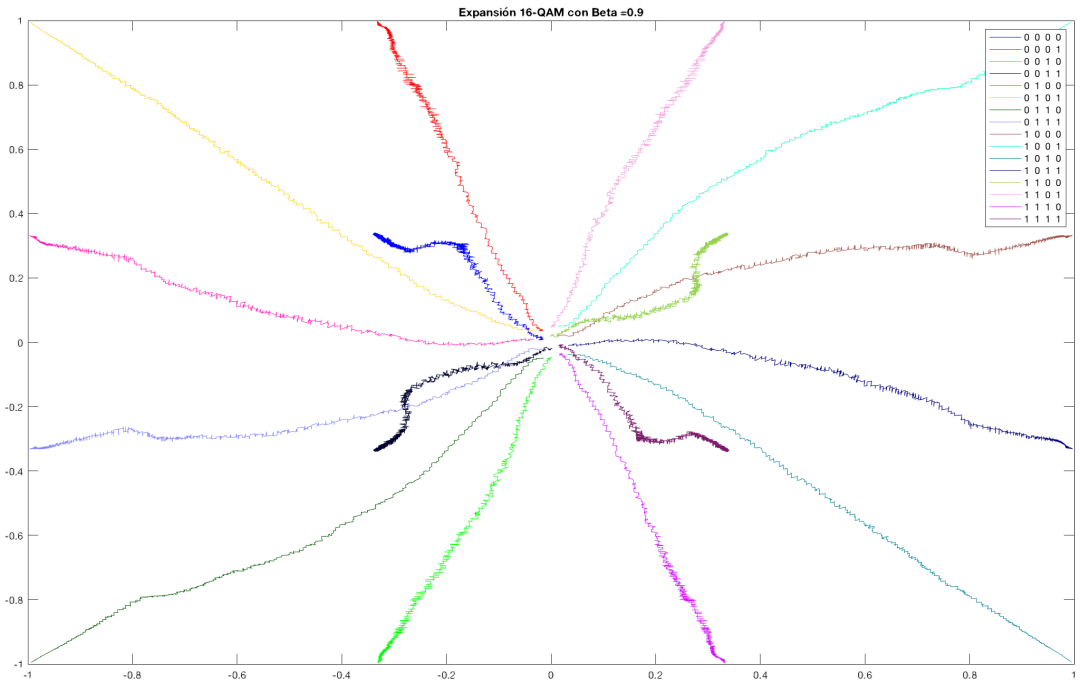

Fig. 7.16. Expansión constelación Beta=0,9 (Trial 2)

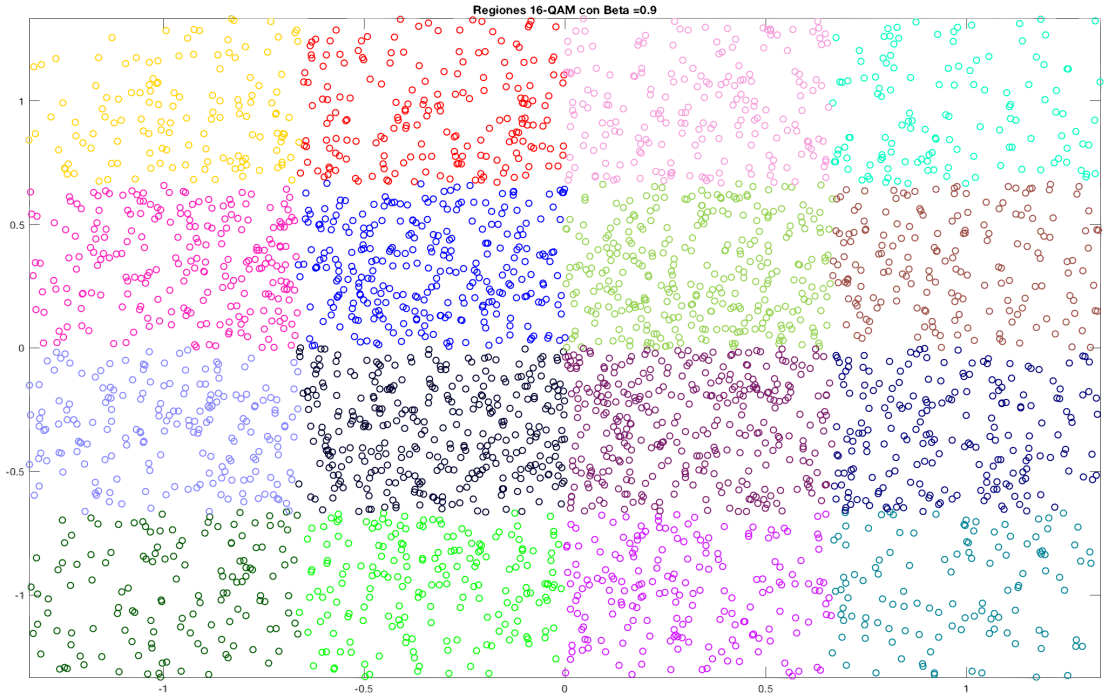

Fig. 7.17. Región de Decisión Beta=0,9 (Trial 2)

A pesar de la obtención de resultados satisfactorios a partir de muestras unitarias, es necesario, como se comentó previamente y con el fin elaborar posteriormente una conclusión razonada, se encuentren los resultados medios de todos los Betas. De esta forma, un

Beta de una determinada simulación que no haya obtenido los resultados habituales por no convergencia debido a factores aleatorios quedará corregido.

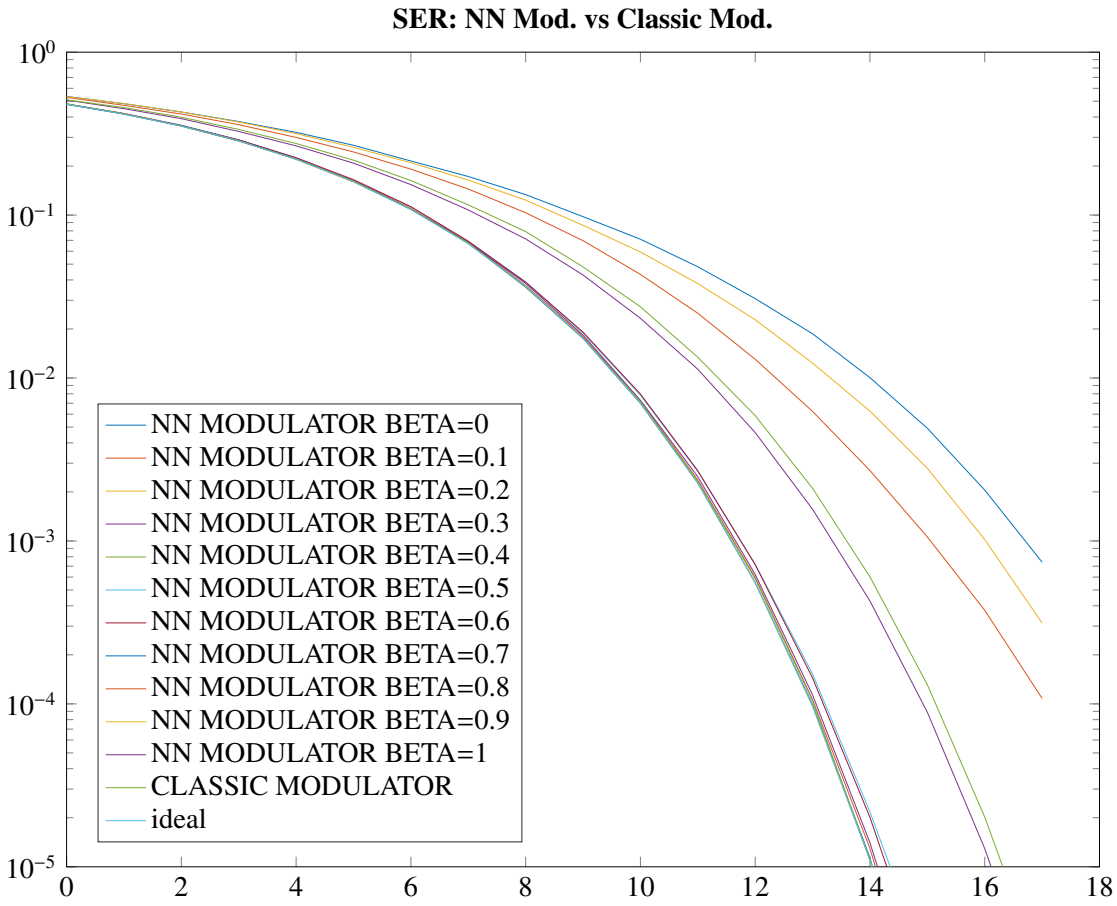

Fig. 7.18. SER 16-QAM Valores medios

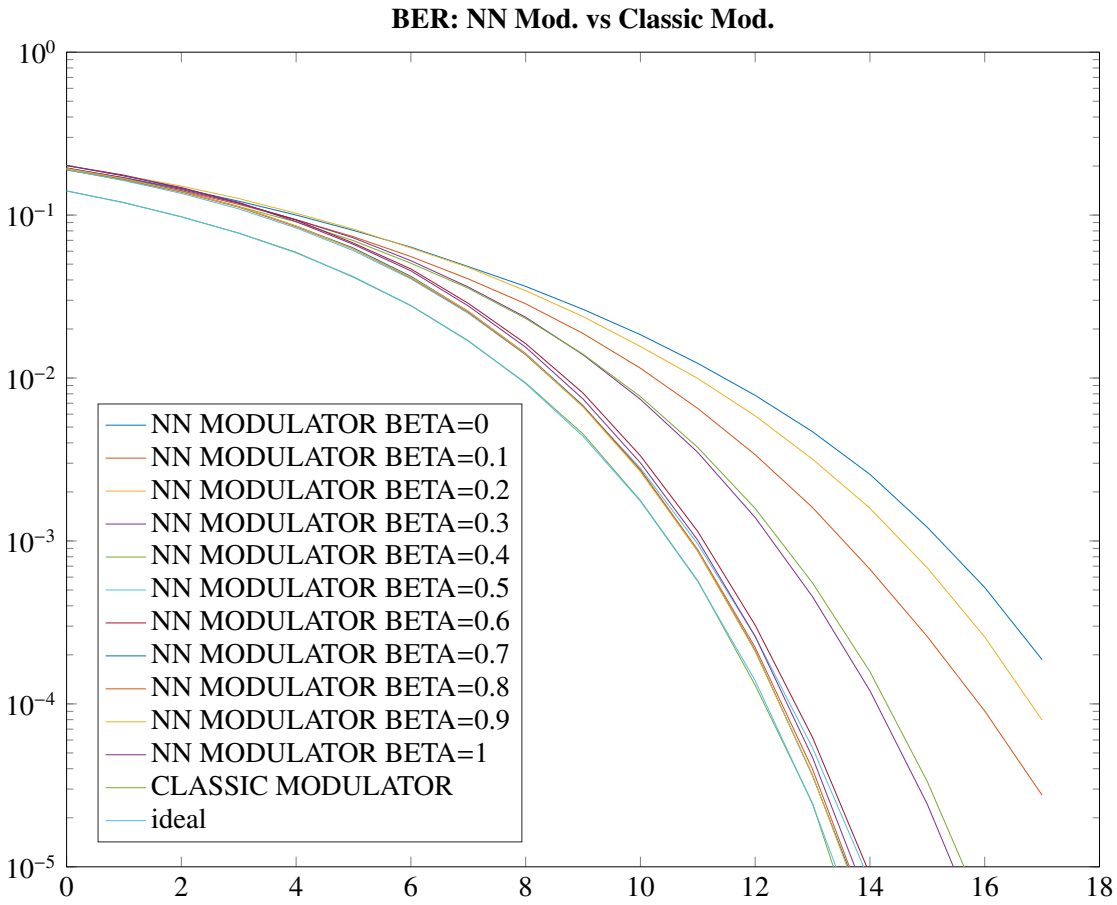

Fig. 7.19. BER 16-QAM Valores medios

7.2. Conclusiones

En primer lugar, este trabajo ha servido para certificar que, mediante la aplicación de un criterio de parada adecuado para la red neuronal, la propuesta obtiene los mismos rendimientos que los sistemas que emplean constelaciones 4-QAM, para todos los Betas. Por tanto, para esta constelación, se consigue relacionar satisfactoriamente el criterio de minimización de la SER con el de distancias ponderadas en todas las ocasiones y para todos los balances de pesos posibles en la función de coste. Esto es así debido a los pocos símbolos de la constelación y a su distancia entre ellos, tal y como se esperaba en la sección 1.2.

Para la constelación 16-QAM, también se consigue determinar una respuesta satisfactoria. Esta constelación, al incluir más símbolos frente a la empleada anteriormente, necesita ser más precisa en la determinación de las regiones de decisión para obtener un rendimiento similar al estandarizado, a partir de una expansión autónoma adecuada.

Tal y como se puede apreciar en las figuras que muestran las tasas medias de error (figuras 7.18 y 7.19) se puede llegar a la conclusión de que existen tres posibles rangos de rendimiento en función del valor de Beta, como se describió en la sección 7.1. Esto es así

tanto en términos de SER como de BER, ya que ambas se encuentran relacionadas (5.1).

En primera instancia, la aplicación de la propuesta con Betas de poco valor ([0,0.2]) implica una relevancia prácticamente máxima de las distancias ponderadas frente al modelo tradicional. Ha quedado demostrado que este uso del sistema no es recomendable, puesto que los rendimientos son muy deficientes frente a los obtenidos de forma tradicional.

En segundo lugar se puede determinar un segundo bloque de valores de Beta ([0.3,0.4]) con los que, aunque la importancia del sistema ponderado sigue siendo mayor para la red neuronal, se va logrando mejorar las prestaciones del modelo propuesto. Sin ser tampoco recomendable su uso bajo estas condiciones, una potenciación leve del peso de las distancias euclídeas otorga al sistema una clara mejoría.

Finalmente, para el último bloque de valores de Beta ([0.5,1]), se obtienen unos rendimientos idénticos al modelo clásico. Esto significa que, partiendo desde este modelo tradicional, el cual está basado en una minimización de la SER a través de las distancias euclídeas entre puntos, se puede lograr un sistema satisfactorio siempre que el peso de estas distancias sea, como mínimo, prácticamente igual que el otorgado a las distancias ponderadas.

En otras palabras, modificando los pesos de la función de coste de la red neuronal desde un equilibrio de ambos, hacia $Beta = 1$, se logra que la propuesta desarrolle las mismas características en cuanto a rendimientos que los modelos que emplean esta constelación usados en las comunicaciones.

Como conclusión, se ha logrado determinar que es factible introducir redes neuronales en las comunicaciones como base para futuros trabajos sobre constelaciones de mayor rango, así como determinar que es posible otorgar cierto peso a las distancias ponderadas (y en consecuencia a la distancia Hamming) sin que ello afecte a los rendimientos del modelo propuesto.

7.3. Presupuesto del Proyecto

Para la realización de este trabajo han sido necesarios diferentes elementos que serán detallados a continuación:

- **Licencia Matlab**: Gratis al ser de uso libre para estudiantes de la Universidad Carlos III de Madrid
- **Licencia Anaconda Distribution**: Gratis al ser de uso libre.
- **Ordenador**: Precio estimado de 1.000 euros, con el que emplear Matlab y Spyder (Scientific Python Development Environment, dentro de Anaconda). Además se emplea para redactar la memoria.

- **Coste de Personal**: Suponiendo un año laboral de 217 días, una jornada de 8 horas y un sueldo de ingeniero junior de 21.000 euros brutos anuales, se puede cobrar en torno a 12 euros/hora. Haciendo una estimación de horas trabajadas al día de 4 para la consecución del proyecto, y como se puede deducir de la Fig.7.20, se ha trabajado en el proyecto (quitando la preparación de la exposición) 162 días. Mediante cálculos evidentes, se obtiene un total de coste de personal de 7.776 euros (más IVA).

Por tanto el presupuesto necesario para la implementación del proyecto puede rondar los 9.000 euros.

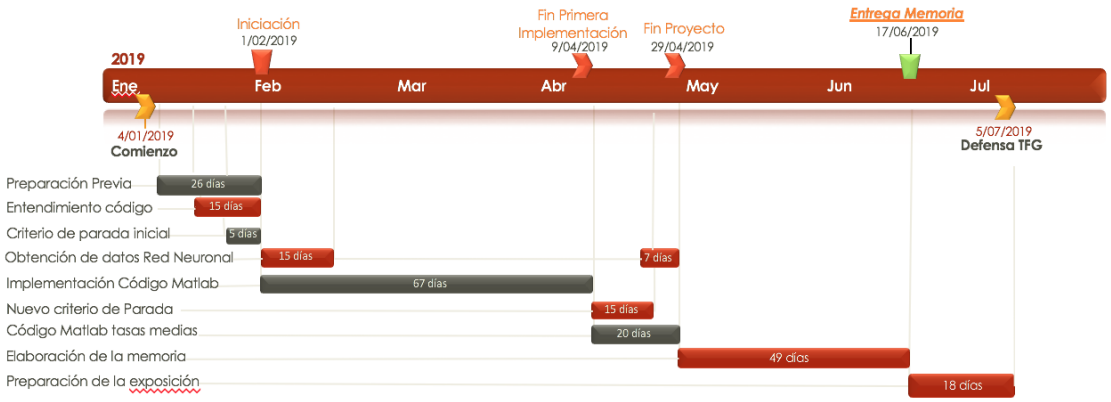

Fig. 7.20. Diagrama de Gantt del Proyecto

8. TRABAJO FUTURO

En este capítulo se tratarán de exponer algunas ideas en referencia a los distintos caminos que podría seguir este proyecto en el futuro. Esto es así debido a que el Machine Learning para comunicaciones detallado en este trabajo es solamente una pequeña pincelada de lo que se puede desarrollar en el futuro.

8.1. Extensión a constelaciones de mayor orden

Evidentemente, al ser este un Trabajo Fin de Grado, no puede llegar a abarcar todas las constelaciones empleadas en la realidad de las comunicaciones. Simplemente se trata de una prueba de concepto, la cual, realmente será de mayor relevancia en la práctica, donde los rendimientos ofrecidos por esta propuesta puedan producir una mejora en las comunicaciones.

Sin embargo, para la aplicación de constelaciones mayores, por ejemplo de orden 256 o 1.024, se necesitará una capacidad de cómputo mucho mayor. Por ello, se recomienda encarecidamente el empleo de computadoras de mayor procesamiento que un ordenador personal, debido a los mayores tiempos de compilación y ejecución durante el análisis de rendimientos, a medida que se incrementa el número de símbolos a transmitir.

Una vez superado este impedimento, sería de gran interés, no solo académico, sino también comercial, el análisis de los resultados en estas constelaciones empleadas en las comunicaciones. A partir de los obtenidos en este trabajo, existe la fuerte posibilidad de que estos sean cada vez más interesantes frente a los del modelo estandarizado actual.

Este hecho supondría una solución alternativa para la mejora de las prestaciones ofrecidas en las comunicaciones, ya que para un cierto rango de Beta, se obtendrán unos rendimientos satisfactorios a partir de la influencia en la red neuronal de ambos modelos en la función de coste.

8.2. Propuesta de nuevos esquemas de codificación

A pesar de que la extensión a constelaciones de mayor orden ya es de por sí una buena línea de desarrollo futuro, existe una de mayor complejidad.

Se trata de proponer nuevos esquemas de codificación. Durante este proyecto se han podido representar los símbolos dado que se parte de un eje bidimensional ($n = 2$). Sin embargo, no se sabe con certeza qué ocurriría si las dimensiones del eje de coordenadas aumentasen, entrando en un hipercubo.

Este simple hecho produce que se pase del escenario de modulación desarrollado du-

rante el proyecto (M símbolos, cada uno de ellos con k bits) a un escenario de codificación. En esta ocasión se lograría tener 2^k patrones de k bits de información, codificados en 2^k codewords de n bits.

BIBLIOGRAFÍA

[1] J. Díaz, *Fundamentos de redes inalámbricas : companion guide*, ép. Academia de Networking de Cisco Systems. Pearson Educación, 2006. [En línea]. Disponible en: https://books.google.es/books?id=u5ohAAAACAAJ.

[2] A. Bateman, *Comunicaciones digitales: diseño para el mundo real*, ép. ACCESO RÁPIDO. Marcombo, 2003. [En línea]. Disponible en: https://books.google.es/books?id=DDLvSzM%5C_hecC.

[3] A. Rodríguez y F. González, *Comunicaciones digitales*. Pearson Educación/Prentice Hall, 2007. [En línea]. Disponible en: https://books.google.com.ec/books?id=0Rn3PQAACAAJ.

[4] R. L. Peterson, D. E. Borth y R. E. Ziemer, *An Introduction to Spread-Spectrum Communications*, 1st. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1995.

[5] F. Seco, J. Carlos Prieto, A. Jiménez y J. Guevara, "Compensation of Multiple Access Interference Effects in CDMA-Based Acoustic Positioning Systems", *Instrumentation and Measurement, IEEE Transactions on*, vol. 63, pp. 2368-2378, oct. de 2014. DOI: 10.1109/TIM.2014.2312511.

[6] J. R. Barry, E. A. Lee y D. G. Messerschmitt, *Pulse-Amplitude Modulation. In: Digital Communication*. Springer, Boston, MA, 2004. DOI: https://doi.org/10.1007/978-1-4615-0227-2_5.

[7] U. Schreiber, "Pulse-Amplitude-Modulation (PAM) Fluorometry and Saturation Pulse Method: An Overview", en *Chlorophyll a Fluorescence: A Signature of Photosynthesis*, G. C. Papageorgiou y Govindjee, eds. Dordrecht: Springer Netherlands, 2004, pp. 279-319. DOI: 10.1007/978-1-4020-3218-9_11. [En línea]. Disponible en: https://doi.org/10.1007/978-1-4020-3218-9_11.

[8] W. T. Webb, *Modern Quadrature Amplitude Modulation: Principles and Applications for Fixed and Wireless Channels*. Piscataway, NJ, USA: IEEE Press, 1995.

[9] L. A. Romero, "Perceptrón Multicapa", [En línea]. Disponible en: http://avellano.fis.usal.es/~lalonso/RNA/introMLP.htm.

[10] K. L. Priddy y P. E. Keller, *Artificial Neural Networks: An Introduction (SPIE Tutorial Texts in Optical Engineering, Vol. TT68)*. SPIE- International Society for Optical Engineering, 2005.

[11] J. Schmidhuber, "Deep Learning in Neural Networks", *Neural Netw.*, vol. 61, n.º C, pp. 85-117, ene. de 2015. DOI: 10.1016/j.neunet.2014.09.003. [En línea]. Disponible en: http://dx.doi.org/10.1016/j.neunet.2014.09.003.

[12] L. Manjarrez, "Relaciones Neuronales Para Determinar la Atenuación del Valor de la Aceleración Máxima en Superficie de Sitios en Roca Para Zonas de Subducción", *ResearchGate*, [En línea]. Disponible en: https://www.researchgate.net/figure/Figura-III4-Capas-de-una-Red-Neuronal-Capa-de-entrada-neuronas-que-reciben-datos-o_fig3_315762548.

[13] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain", *Psychological Review*, pp. 65-386, 1958.

[14] W. Commons, *File:Perceptrón 5 unidades.svg — Wikimedia Commons, the free media repository*, [Online; accessed 16-June-2019], 2016. [En línea]. Disponible en: %5Curl%7Bhttps://commons.wikimedia.org/w/index.php?title=File:Perceptr%5C%C3%5C%B3n_5_unidades.svg&oldid=223351597%7D.

[15] J. L. McClelland, D. E. Rumelhart, P. R. Group et al., "Parallel distributed processing", *Explorations in the Microstructure of Cognition*, vol. 2, pp. 216-271, 1986.

[16] D. E. Rumelhart, R. Durbin, R. Golden e Y. Chauvin, "Backpropagation", en, Y. Chauvin y D. E. Rumelhart, eds., Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1995, cap. Backpropagation: The Basic Theory, pp. 1-34. [En línea]. Disponible en: http://dl.acm.org/citation.cfm?id=201784.201785.

[17] M. Pereyra, "Maximum-a-Posteriori Estimation with Bayesian Confidence Regions", *SIAM Journal on Imaging Sciences*, vol. 10, n.º 1, pp. 285-302, 2017. DOI: 10.1137/16M1071249. eprint: https://doi.org/10.1137/16M1071249. [En línea]. Disponible en: https://doi.org/10.1137/16M1071249.

[18] W. Hong, K. Baek, Y. Lee, Y. Kim y S. Ko, "Study and prototyping of practically large-scale mmWave antenna systems for 5G cellular devices", *IEEE Communications Magazine*, vol. 52, n.º 9, pp. 63-69, sep. de 2014. DOI: 10.1109/MCOM.2014.6894454.

[19] W. Commons, *File:16QAM Gray Coded.svg — Wikimedia Commons, the free media repository*, [Online; accessed 16-June-2019], 2017. [En línea]. Disponible en: %5Curl%7Bhttps://commons.wikimedia.org/w/index.php?title=File:16QAM_Gray_Coded.svg&oldid=260094091%7D.

[20] P. Vaidyanathan, S. Phoong e Y. Lin, *Signal Processing and Optimization for Transceiver Systems*, ép. Signal Processing and Optimization for Transceiver Systems. Cambridge University Press, 2010. [En línea]. Disponible en: https://books.google.es/books?id=Icq1bNpr3xUC.

ANEXO

The aim of this project is to show how Machine Learning can affect and contribute to communications, providing more efficient Modulations. Therefore, the final purpose is trying to achieve better performances than the standard modulation, by using neuronal networks and other model rather than minimazing the Symbol Error Rate (SER) in the implementation.

The philosophy of this Final Degree Work is based on the behaviour behind Neuronal Networks, thanks to their ability of developing autonomously from a function written by the developers. In this aspect, the system will accomplish different training phases, in which, starting with some input data, the neuronal network will converge, giving back at the end the constellation obtained.

On the other hand, another point of the project tries to make a comparison between the results of this proposal and the ones of the traditional and standardized communications systems.

Until now, systems have usually based their judgment in minimazing the SER, obtained only by Euclidean distance. However, this work tries to achieve some balance between the standard and another model based on weighted distances (through Hamming distance). For this reason, the project implies researching to conclude if it is possible to improve the traditional system.

The proposed model is only applicable to squared-form constellations (such as 4-QAM or 16-QAM, and excluding others like 8-QAM or 32-QAM due to their form), because of the facilities with the calculation of distances involved.

The expected results of this work are that in constellations with few symbols (4-QAM), and therefore more distance between them, the differences with the standardized model would be insignificant in terms of performance.

However, the main constellation of the project will be only 16-QAM, due to processing and compute capacity of the personal computer. Applying this constellation, it is reasonable to think that the performance would be decent starting at some point, but not the whole range of results. If the project turns out successful, it would mean that it is possible to obtain an alternative solution for communication systems.

Nevertheless, if the project shows that it is not possible to achieve that decent performances, the work would be also valuable. It would determine that the use of weighted distances, obtained through the bases of Hamming distance and Gray codification, in the communication process, would be critical and inadvisable its use in the development of the Modulator.

The primitive idea is based on using a Neuronal Network to develop the constellation

from the coordinates origin. It is developed by a function, written in Python, where a graph will be defined, and inside it would be some instructions to be executed during the process.

First of all, inside the graph, it is needed to define some Trainable Variables, which will be modified at the trainable phase. Thanks to this, the bits received as inputs to the neuronal network turn out symbols of the constellation as outputs.

After that, it is defined the different operations, calculated during the execution (for example the instructions to calculate both distances), that will take the neuronal network. From the use of tensors, the Euclidean distance and the weighted one will be determined, but to accomplish that, it is necessary to reserve space in memory for some auxiliary variables used during the process.

Second of all, it is defined an auxiliary function, which is called from the main function, to carry out the modulation process, and going back to the graph when the execution has been through all the layers of the neuronal network, providing at the end the Tensor obtained. In order to get the appropriate range of results, it is necessary to apply a determined non-linearity function in each layer, the hyperbolic tangent. This function limits the results between ± 1, letting the constellation to develop correctly.

Besides it is defined another auxiliary function, also called from the main function, to get the different pairwise matrices in form of Tensors, and their respective minimum distance of the Euclidean system and the weighted one.

One of the basic aspects of a Neuronal Network implies to define a Cost Function. This basic point consists on letting the neuronal network to learn the behaviour behind the inputs received, by minimizing this determined Cost Function. Therefore, it would be modified in function of the specific values given to the weights. The parameter to do this, called Beta, it is used to determine how important would be each kind of distance during the execution, obtaining a wide range of performances between each result set.

After the modification, already explained, of the Cost Function, the trainable variables have to be modified. Generally, this process is done by minimazing the Cost Function. However, this project implies the maximization of the distances between symbols of the constellation. Hence, a small trick needs to be involved in this situation. By changing the sign of the Cost Function, and then minimazing it, the appropriate result is achieved.

At some point, the Neuronal Network will stop working. In order to complete a successful execution, and this is a key aspect of the work, it is necessary to implement the right stop algorithm. At first implementation, the stop criterion was based on a maximum number of iterations. That was decided in that way due to the expectations that the neuronal network would develop in a linear way. Therefore, the results expected for the expansion of the contellation were that the points would move directly to their maximizing position, without any kind of loops or curves.

Once the instructions of the neuronal network are defined, the basic parameters of

the Neuronal Networks have also to be defined. Some of them are the order of the constellation, the number of layers of the system, the number of dimensions (axis) or the non-linearities applied to each layer. The system proposed was very convenient for the user as it was only needed a call to the function that acted as a black box to work. In each iteration, the trainable variables are modified, letting the neuronal network to keep learning autonomously. Besides this, the rest of parameters used inside the function are saved in a specific dictionary for neuronal networks, so they can be used in further iterations. Also the results in terms of distances and points of the constellation are kept to be used in the graphic representations.

Finally, once the session has been initialized and executed according to the stop algorithm previously defined, the result sets are prepared to be used in Matlab. This programming language is used to show graphically the constellations obtained thanks to the neuronal network and the analysis between the proposed model and the standardized one. The structure of this analysis had three steps. The first one develops continuously the points of the constellation obtained during the execution of the neuronal network. Thanks to this, the user is capable of viewing the path took by the points during the maximizing process and compare it to the form of the ideal constellation (grid shape).

In order to simplify the understanding of the first step, the second was created. In this occasion, code to show the different regions of decision was implemented. Thanks to this, the user can determine the point selected according to the specific area where is located. As the results are likely to be different, in many occasions, to the traditional grid shape, this code will show the shapes selected by the neuronal network allowing to compare them to this ideal model.

The final step in Matlab would mean the development of the necessary code to show the Symbol Error Rate (SER) and the Bit Error Rate (BER) of the results. As this two error rates are related, both results will be similar always. In order to avoid problems related to the randomness of the neuronal network, code to show the average error rate for both of them is also implemented. Thanks to this, the performance for each value of Beta will be compared in the same graphic to the ideal and also to the traditional modulation system, letting the user to obtain a clear decision about the utility of the proposal.

This first trial of the project showed that the expectations of the 4-QAM constellation were fulfilled. There was no difference between the performance of the proposed system and the classic model. That meant that the next step was do the same with the 16-QAM constellation.

The results for this one were not expected, as the performance obtained was far from the standardized model. In order to correct the problem, the 4-QAM constellation came back as it required less compute time. After the execution of several simulations, an anomalous results was found in two different trials.

This fact could only have one motive, the stop algorithm was not appropriate, due to sometimes the neuronal network need more iterations before converging, and that num-

ber of iterations was higher than the maximum one established. Therefore, a brand new criterion had to be defined.

The new algorithm was more complex, due to the maximum number of iterations allowed was changing dynamically in function of the concrete necessities of the execution. The base of this new one is the value of the Cost Function. This value would be examined when a certain percentage (10 %) of the current maximum value of iterations is accomplished, and then would be save in a stack, with capacity for three values.

When the number of iterations reach a higher percentage (70 %) and the value of the Cost Function is greater than 2 out of three values of the stack (so it means that the constellation points are still being maximized), the maximum value of allowed iterations would increase a 2 %. In this way, this new criterion is more flexible and adaptive for the specific necessities, allowing the neuronal network to converge in any case.

This new algorithm had to be modified in the Python code, but as the stop criterion is defined outside the neuronal network, it had not to be modified, as it has always worked properly. The changes took little time to be implemented, as it only meant the sentences inside 'if' statements and the creation of the stack to save the values of the Cost Function received at runtime from the neuronal network.

After this changes, new simulations were needed to approve the new mechanic of the algorithm. Due to the good feelings with this new criterion (it changes in function of the value of the Cost Function, which is key for the convergence of a neuronal network), only trial with 16-QAM were required. It was supposed that the smaller constellation, 4-QAM, would react properly in any case of execution. However, after the disappointment of the first attempt with the 16-QAM constellation, it was expected to work correctly from now and obtain finally the expected results.

At this time, the results were decent, so could analyze them and take the proper conclusions. Three big blocks of solutions, in function of the value of the Beta used during the execution, were built. First of all, applying a low range of Beta [0,0.2] values to the system, the results have shown that it is not recommendable to do such thing, as the performances obtained are far from good in comparison to the traditional model.

By incrementing the value of Beta, until values rounding 0.4, it also increases the performance of the neuronal network. The result of the use of the system with this range of Beta can not be considered appropriate despite of the improvement achieved. Therefore, the performance of this block is between the ideal one and the one of the first block.

Finally, for the rest of the possible values of Beta the aim of the project is achieved. From the maximum value of Beta (1), which means that the neuronal network applies the traditional model by using the Euclidean distance, to values rounding 0.5, the performances obtained for every Beta are equal between them and equal to the classic modulator one and to the ideal one.

This fact proves that balancing the weights of the two models proposed (Euclidean

and weighted distance), but always considering the minimazing of the SER model more important turns out a successful experiment.

As a conclusion to this Final Degree Work, it is reasonable to think that this proof of concept adequate to keep developing and testing through this idea of neuronal networks and weighted distances applied to real communications.

In this way there are two possibles paths of development for future researchings. The first one would mean to increase the order of the constellation used, approaching to the ones that are applied in the standardized communications (64-QAM,etc). The expected results of this future project are likely to be also satisfactory as the neuronal network could obtain more efficient results.

The other way for future aproximations to the idea consists of proposing new codification schemes. Although this path would be much more complicated, it also could provide larger profits by increasing the dimensions of the coordinates axis, obtaining a hypercube form. This simple fact leads to a change, from a modulation scenario (M symbols of k bits) to a codification one, where there are much more combinations available. In this way, this proposal can be considered as a true experiment for a future implementation of the system described in this project.